E.T.S. de Ingeniería Industrial, Informática
y de Telecomunicación

Dispositivo de medición y monitorización remota para centrales solares fotovoltaicas de autoconsumo con baterías con Arduino

Grado en Ingeniería
en Tecnologías Industriales

Trabajo Fin de Grado

Autor: Xabier Bezunartea Oroz

Director: Pablo Sanchis Gúrpide

Pamplona, 10 de junio de 2021

upna
Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

RESUMEN

En el presente proyecto se ha realizado el diseño de un dispositivo de monitorización remota para instalaciones solares fotovoltaicas de autoconsumo con baterías. Esta modalidad de autoconsumo es el presente y futuro del sector de la generación eléctrica renovable y contribuye a diario a reducir la dependencia energética y el gasto de gran cantidad de hogares y empresas.

Este dispositivo pretende ofrecer una solución sencilla y económica para los usuarios que quieran monitorizar sus instalaciones fotovoltaicas de autoconsumo. Se trata de un proyecto IoT (Intenet of Things) que toma las medidas necesarias para monitorizar la instalación y publica los resultados en un portal web a disposición del usuario.

Se ha diseñado un portal web propio que almacena y publica los datos obtenidos por el dispositivo. Este portal web se encuentra disponible en la siguiente dirección.

www.mienergiafotovoltaica.com

ABSTRACT

In the present dissertation, has been designed a remote monitoring device for self-consumption solar photovoltaic installations with batteries. This type of self-consumption is the present and future of the renewable electricity generation sector and contributes daily to reduce the energy dependence and electric expenditures of a large number of homes and businesses.

This device aims to offer a simple and economical solution for users who want to monitor their self-consumption photovoltaic installations. It is an IoT (Internet of Things) project that takes the necessary measures to monitor the installation and publishes the results on a web portal available to the user.

The project includes a web portal that stores and publishes the data obtained by the device. This web portal is available at the following address.

www.mienergiafotovoltaica.com

LISTA DE PALABRAS CLAVE

Arduino

Microcontrolador

AtMega2560

Sensor

Placa

Monitorización

Generación

Consumo

Autoconsumo

Excedentes

Importación

Portal web

ACRONIMOS

IoT	"Internet de las cosas" *(Internet of Things)*
IDE	Entorno de Desarrollo Integrado *(Integrated Development Environement)*
CSFV	Central Solar Fotovoltaica
DC	Corriente Continua *(Direct Curruent)*
AC	Corriente Alterna *(Alternating Current)*
LPF	Filtro Paso Bajo *(Low Pass Filter)*
ADC	Convertidor Analógico-Digital *(Analog-Digital Converter)*
BD	Base de Datos
IP	Interfaz de Programación

ÍNDICE

RESUMEN 3
ABSTRACT 4
LISTA DE PALABRAS CLAVE 5
ACRONIMOS 5
ÍNDICE DE FIGURAS 9
ÍNDICE DE TABLAS 11
CAPÍTULO 1: INTRODUCCIÓN 12
1.1. Antecedentes 12
1.2. Objeto 12
1.3. Justificación e interés comercial 12
1.4. Sistemas fotovoltaicos. Topologías 13
1.5. Caracterización de una instalación fotovoltaica de autoconsumo. 13
CAPÍTULO 2: MEDICIÓN 16
2.1. Elección de la tecnología y entorno de desarrollo 16
2.1.1. Entorno de desarrollo y hardware 16
2.1.2. IDE Arduino 16
2.1.3. Microcontroladores Atmel y placas de desarrollo. 16
2.2. Medida de la corriente AC 17
2.2.1. Introducción 17
2.2.2. Elección del método de sensado 17
2.2.3. Acomodo de la señal de salida del sensor 19
2.2.4. Señal de salida de los sensores SCT-013-0xx 19
2.2.5. Circuito de offset 20
2.2.6. Circuito amplificador no inversor 21
2.2.7. Adecuación de la señal de salida de los sensores SCT-013-0xx. 23
2.2.8. Programación de la lectura del sensor de corriente AC 24
2.3. Medida de la tensión AC 28
2.3.1. Introducción 28
2.3.2. Elección del método de sensado 28
2.3.3. Conexión con la placa 29
2.3.4. Ajuste de la señal de salida del sensor y calibración. 30
2.4. Medida de la corriente DC 35
2.4.1. Introducción 35
2.4.2. Elección del método de sensado 36
2.4.3. Adecuación de la señal de salida del sensor 36

2.4.4. Conexión con la placa 38
2.4.5. Programación 38
2.5. Medida de la tensión DC 40
2.5.1. Introducción 40
2.5.2. Elección del método de sensado 40
2.5.3. ADS1115 42
2.5.4. Circuito de medida. Conexión con la placa 43
2.5.5. Programación 43
CAPÍTULO 3: CÁLCULOS Y PROGRAMACIÓN 46
3.1. Introducción 46
3.2. Cálculos 46
3.2.1. Potencia y energía en las baterías, inversor y acometida. 47
3.2.2. Generación fotovoltaica 49
3.2.3. Excedentes y compra de energía 49
3.2.4. Consumo eléctrico 50
3.2.5. Energía autoconsumida 50
3.2.6. Porcentaje de autoconsumo y cobertura de la demanda 51
3.2.7. Carga de las baterías 51
3.3. Gestión de excedentes 52
3.4. Avisos y alarmas 53
3.4.1. Carga de las baterías 53
3.4.2. Alarma por desequilibrio de tensión entre las fases del sistema 54
CAPÍTULO 4: MONTAJE Y COLOCACIÓN DEL DIPOSITIVO 56
4.1. Montaje experimental 56
4.2. Montaje definitivo 56
4.3. Colocación 58
CAPÍTULO 5: PUBLICACIÓN DE LOS DATOS OBTENIDOS 61
5.1. Introducción 61
5.2. Dominio 61
5.3. Alojamiento en servidor web 61
5.4. Base de datos 62
5.4.1. Diseño de la base de datos 62
5.4.2. Creación de la base de datos 63
5.5. Envío de datos 64
5.6. Recepción y almacenamiento de datos 67
5.7. Diseño de la página web. Wordpress 69

5.7.1. Instalación de wordpress 69
5.7.2. Colibri Page Builder 70
5.7.3. WpDataTables 70
BIBLIOGRAFÍA 75
ANEXOS 76
ANEXO I: Programación para autoconsumo trifásico con baterías 76
ANEXO II: Programación para autoconsumo monofásico con baterías 89
ANEXO III: Programación para autoconsumo trifásico sin baterías 100
ANEXO IV: Programación para autoconsumo monofásico sin baterías 110
ANEXO V: Plano de montaje del medidor 117

ÍNDICE DE FIGURAS

Fig 1 Autoconsumo fotovoltaico sin baterías 14
Fig 2 Autoconsumo fotovoltaico con baterías 15
Fig 3 Placa de expansión para AtMega2560 17
Fig 4 Funcionamiento sensores SCT-013-0xx 18
Fig 5 Sensor SCT-013-030 19
Fig 6 Salida de los sensores SCT-013 20
Fig 7 Circuito de offset 20
Fig 8 Salida del circuito de offset 21
Fig 9 Circuito amplificador no inversor 22
Fig 10 Salida del circuito amplificador no inversor 23
Fig 11 Circuito de adecuación de la señal para los sensores SCT-013-0xx 23
Fig 12 Salida acomodada de los sensores SCT-013 24
Fig 13 Gestor de librerías 24
Fig 14 Pinza ampermétrica comercial 25
Fig 15 Medida de la corriente 26
Fig 16 Calibración del sensor 26
Fig 17 Resultados de la calibración 27
Fig 18 Divisor de tensión 28
Fig 19 Sensor de tensión AC 29
Fig 20 Conexión con la placa: sensor de tensión AC 30
Fig 21 Calibración del sensor de tensión AC 31
Fig 22 Filtro Paso Bajo de primer orden 32
Fig 23 Circuito de sensado y filtro 32
Fig 24 Resultados tras filtrado 33
Fig 25 Filtrado de la señal de salida del shunt 37
Fig 26 Simulación EMI y filtrado 38
Fig 27 Divisor de tensión 41
Fig 28 Salida del divisor de tensión 42
Fig 29 ADS1115 43
Fig 30 Código de colores para indicador de carga de las baterías 53
Fig 31 Montaje experimental 56
Fig 32 Detalle de la técnica de soldadura 56
Fig 33 Montaje definitivo sin tapa 57
Fig 34 Montaje definitivo con tapa 57
Fig 35 Detalle de las conexiones de los sensores de voltaje y corriente AC 58
Fig 36 Conexión del shunt y medida de voltaje de las baterías 59
Fig 37 Foto detalle del sensor SCT-013-030 59
Fig 38 Conexiones del dispositivo 60
Fig 39 Panel de administración del servidor web 62
Fig 40 Creación de bases de datos 63
Fig 41 Propiedades de la tabla 64
Fig 42 Panel de administración de Wordpress 70
Fig 43 Página principal de www.mienergiafotovoltaica.com 70
Fig 44 Tabla de resultados 71
Fig 45 Tabla de datos históricos 71
Fig 46 Gráfico de evolución diaria 72

Fig 47 Gráfico de totales diarios 72
Fig 48 Gráfico de generación por horas 72
Fig 49 Gráfico de consumo por horas 73
Fig 50 Gráfico de autoconsumo por horas 73
Fig 51 Gráfico de vertidos por horas 73
Fig 52 Gráfico de importación por horas 73
Fig 53 Gráfico de la evolución histórica del porcentaje de autoconsumo 74
Fig 54 Gráfico de la evolución histórica de la cobertura de la demanda 74

ÍNDICE DE TABLAS

Tabla 1 Comparación microcontroladores Atmel 17
Tabla 2 Gama de sensores SCT-013 19
Tabla 3 Ganancias internas del ADS1115 44
Tabla 4 Tabla para almacenar datos en base de datos 63

CAPÍTULO 1: INTRODUCCIÓN

1.1. Antecedentes

Este proyecto de final de carrera lo he llevado a cabo durante mi participación en el programa de prácticas curriculares de la UPNA, durante el cual he estado trabajando con el departamento de ingeniería de la empresa Energías Fotovoltaicas de Navarra (FOTONA).

Durante mi estancia en FOTONA he participado en el desarrollo de instalaciones solares fotovoltaicas de autoconsumo tanto en el ámbito industrial como residencial. Al finalizar los proyectos, he observado que los usuarios piden tener acceso a los datos de generación, consumo, autoconsumo y demás a través de algún tipo de portal web o aplicación. He podido observar también que los dispositivos comerciales que existen en el marcado son costosos y en ocasiones los clientes deciden prescindir de ellos, perdiendo la posibilidad de monitorizar su propia instalación.

Actualmente, el sector del autoconsumo eléctrico, especialmente mediante generadores solares fotovoltaicos es un sector al alza. Cada día más empresas y hogares implementan un generador fotovoltaico sobre su cubierta con el fin de producir la energía eléctrica que demandan, al menos parte de ella, y reducir notablemente los costes en su tarifa eléctrica.

1.2. Objeto

Ante esta situación, he decidido diseñar un dispositivo de monitorización remota que permita visualizar en tiempo real y de manera retrospectiva a través de un portal web propio los datos de generación, consumo, autoconsumo y demás datos de interés de cualquier instalación fotovoltaica de autoconsumo con o sin acumulación de energía.

El objeto de este proyecto, más allá del desarrollo del dispositivo radica en facilitar la comprensión de unos datos de carácter técnico a usuarios no necesariamente formados en la materia. Además, se trata de diseñar una opción que se diferencie tanto en prestaciones como en precio frente a las opciones comerciales actuales.

El dispositivo diseñado deberá ser también fácilmente instalable para no perturbar en exceso la naturaleza de las instalaciones fotovoltaicas presentes y futuras donde se instale.

1.3. Justificación e interés comercial

Uno de los retos a los que se enfrenta el sector del autoconsumo fotovoltaico es el hecho de acercar la tecnología a los usuarios y facilitar su comprensión. Con esta finalidad, los grandes fabricantes de equipos para instalaciones fotovoltaicas ofrecen sistemas de monitorización compatibles con sus propios productos.

Actualmente existe una oferta limitada de equipos con funciones similares a las descritas anteriormente. Estos equipos pueden ser de dos tipos. En primer lugar, existen dispositivos de conexión a internet, los cuales recogen los datos tomados por el inversor, regulador de carga y demás equipos y los publican en un portal web propio del fabricante. Estos dispositivos no son medidores, por lo que deben ser necesariamente compatibles con los demás equipos de la instalación, reduciendo notablemente el abanico de opciones comerciales. En segundo lugar, existen dispositivos de medida directa, los cuales publican los datos obtenidos en un portal web. Sin embargo, el precio de mercado de estos dispositivos es muy elevado, por lo que su implementación en instalaciones residenciales o industriales de pequeña potencia es poco frecuente.

Dicho esto, se detecta el nicho de mercado para un dispositivo de medida directa y de bajo coste. Al tomar sus propias medidas, este dispositivo no deberá comunicarse con el resto de equipos.

1.4. Sistemas fotovoltaicos. Topologías

Las instalaciones solares fotovoltaicas se pueden enmarcar en dos categorías: instalaciones de conexión a red e instalaciones aisladas.

- Centrales solares fotovoltaicas de conexión a red

Las instalaciones de conexión a red son aquellas que están interconectadas a la red eléctrica. Entre las instalaciones de conexión a red se diferencian dos tipos de instalaciones.

En primer lugar, destacan las instalaciones puramente generadoras, cuyo objetivo es la generación de energía eléctrica para abastecer a la red. Estas instalaciones suelen ser de gran tamaño y se ubican en grandes superficies de terreno, sobre el suelo y pueden implementar algún tipo de seguimiento solar para maximizar la generación de energía. Éstas carecen de interés para este proyecto ya que la monitorización de este tipo de centrales se realiza de otra manera y con finalidades diferentes.

En segundo lugar, existen instalaciones fotovoltaicas de autoconsumo. Estas instalaciones se encuentran próximas a los puntos de consumo, generalmente sobre la cubierta de la industria u hogar consumidor, y su objetivo principal es abastecer de energía eléctrica a los usuarios. Las instalaciones fotovoltaicas de autoconsumo sobre cubierta son actualmente la mejor opción para reducir la dependencia energética de los consumidores que cuentan con el espacio para su construcción. Son instalaciones de conexión a red ya que la energía producida por la instalación que no se consume directamente en ese momento se vierte a la red eléctrica (para ser compensada por la comercializadora) y la energía restante para cubrir el consumo del usuario se importa desde la propia red eléctrica, cuando es necesario.

Estas instalaciones de autoconsumo pueden implementar o no sistemas de acumulación de energía eléctrica. Actualmente no es lo habitual, pero en un futuro cercano todas las instalaciones de autoconsumo implementaran baterías con el fin de almacenar la energía sobrante producida y consumirla “a coste cero” fura de las horas de producción solar.

Esta topología es para la cual se va a diseñar el dispositivo, ya que es en este tipo de instalaciones donde los usuarios requieren acceso a sus datos de forma sencilla y asequible para sus bolsillos.

- Centrales solares fotovoltaicas aisladas

Las centrales solares fotovoltaicas aisladas son aquellas que no se encuentran conectadas a la red eléctrica.

Estas instalaciones pueden ser de electrificación para hogares que no cuentan con conexión a la red; en estos casos implementan siempre sistemas de almacenamiento de energía eléctrica (baterías). Además, existen también instalaciones de bombeo fotovoltaico utilizadas para bombear agua siempre que haya generación eléctrica.

Estas instalaciones tienen poco potencial en el mercado y son más sencillas tecnológicamente que las de autoconsumo, por lo que no se tendrán en cuenta a la hora de diseñar el dispositivo.

1.5. Caracterización de una instalación fotovoltaica de autoconsumo.

En primer lugar, se procede a describir detalladamente las instalaciones de autoconsumo, con y sin baterías, puesto que estas serán objeto de este proyecto. A continuación, se describen los componentes con conforman este tipo de instalaciones desde el generador fotovoltaico hasta la conexión con la red eléctrica, en orden.

- Generación y almacenamiento de energía. Parte DC.
  - Paneles fotovoltaicos. Los módulos fotovoltaicos conforman el generador eléctrico que convierte la energía solar en electricidad mediante el efecto fotovoltaico.
  - Cuadro de protecciones DC. El cableado de corriente continua y los paneles se protegen mediante fusibles.
  - Regulador de carga y baterías. El almacenamiento de energía se realiza en DC, como etapa previa a la conversión de la energía. El regulador de carga carga las baterías cuando la generación es mayor que la demanda y extrae carga de las mismas cuando la demanda supera a la generación. Estos equipos no son imprescindibles y solo se encuentran en instalaciones de autoconsumo con baterías.
- Etapa de conversión de la energía. Inversión DC/AC
  - Inversor. El inversor es el equipo central de la instalación y se encarga de convertir la energía DC generada por el generador en alterna para su consumo y/o inyección a la red.
- Etapa AC. Consumo y conexión con la red.
  - Cuadro de protecciones AC. El cableado por el que fluye la energía generada a la salida del inversor se protege frente a cortocircuitos y sobretensiones; mediante un interruptor diferencial y un interruptor magnetotérmico normalmente.
  - Inyección en red interior. La generación eléctrica se vierte en la red interior del consumidor para ser aprovechada directamente. Esta conexión se realiza directamente en el cuadro de protecciones de la instalación.
  - Contadores. Toda instalación consumidora tiene un contador bidireccional de consumo y generación, encargado de cuantificar la energía intercambiada con la red en ambos sentidos (importación y vertidos).
  - Acometida. La acometida es la conexión física entra la instalación interior y la red eléctrica.

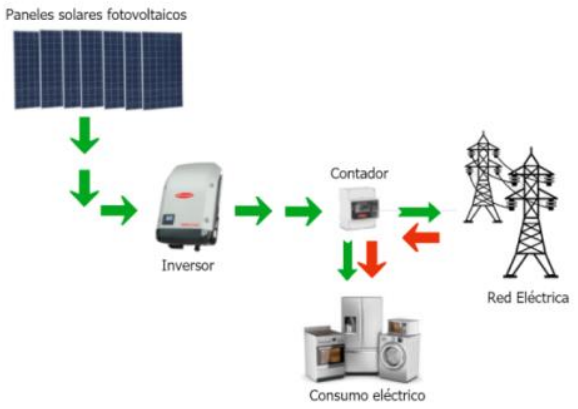


Fig 1 Autoconsumo fotovoltaico sin baterías

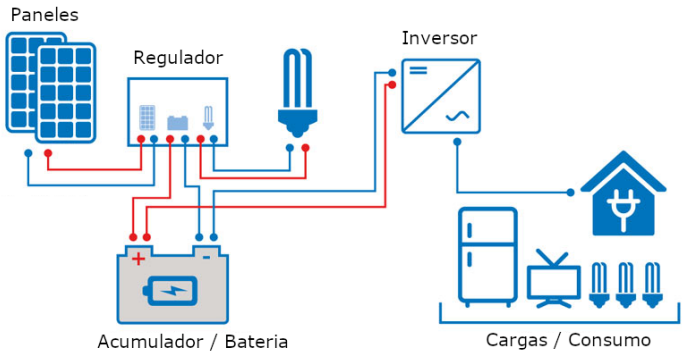


Fig 2 Autoconsumo fotovoltaico con baterías

Puesto que las instalaciones de autoconsumo con baterías son más complejas que aquellas que no almacenan energía, el diseño se realizará para las instalaciones que sí incorporan baterías, siendo este igualmente válido para las otras.

Para caracterizar una instalación fotovoltaica de autoconsumo con baterías se deben tomar las siguientes medidas:

1. Medida de la energía transformada por el inversor.

Los inversores suelen implementar un gran número de entradas (MPPT), por lo que esta medida se tomará siempre a la salida del inversor, ya que en este punto el cableado es más sencillo. Se deberá medir la tensión y la corriente a la salida del inversor fotovoltaico.

En las instalaciones sin baterías esta medida coincidirá con la energía generada en cada momento. Cuando exista almacenamiento, esta medida complementará a la tomada en el cableado de las baterías para calcular la generación eléctrica.

2. Medida de la energía almacenada o cedida por las baterías.

En caso de existir sistemas de acumulación de la energía eléctrica, baterías, se deberá medir en todo momento la energía cedida o almacenada en estas. Para ello, se medirá la tensión en bornes de la batería y la corriente que circula por el cableado de las baterías.

3. Medida de la energía intercambiada con la red eléctrica.

Finalmente, se medirá la energía intercambiada (cedida o importada) con la red eléctrica. Esta medida, junto con las otras obtenidas anteriormente, permitirá cuantificar el consumo total en la instalación, la energía autoconsumida y los excedentes o la energía importada.

Para ello, se medirá la tensión y corriente en cualquier punto entre el cuadro de protecciones y la acometida eléctrica. Por comodidad, esta medida se tomará en bornes del contador.

CAPÍTULO 2: MEDICIÓN

2.1. Elección de la tecnología y entorno de desarrollo

2.1.1. Entorno de desarrollo y hardware

En este primer apartado del bloque de medición se va a tratar de justificar la elección del entorno de programación y del circuito integrado escogido para la realización del proyecto descrito.

2.1.2. IDE Arduino

El entorno de desarrollo escogido para la escribir, probar y ejecutar el código necesario para la solución del problema es el IDE (Integrated Development Environment) de Arduino®. Se trata de un entorno de desarrollo de programas compatible con Linux, Windows y Mac OS.

El IDE de Arduino es una aplicación de software que facilita a programadores el desarrollo de todo tipo de software para ser ejecutado por los microcontroladores Atmel y las placas de desarrollo o circuitos integrados basados en estos microcontroladores.

Arduino es un entorno de desarrollo ampliamente utilizado por varias razones. Entre ellas destaca el hecho de que se trate de software libre, bajo la licencia Pública General de GNU, por lo que cualquier individuo puede hacer uso, manufacturar y distribuir hardware y software compatible con dicho IDE. Gracias a esto, existen infinidad de controladores, placas y componentes compatibles con esta herramienta, además de gran cantidad de información y de códigos abiertos al alcance de cualquiera.

El lenguaje de programación utilizado por el IDE de Arduino es C++, uno de los lenguajes más conocidos y utilizados en la industria y otros ámbitos.

Por todo ello, trabajar con el IDE de Arduino y con hardware y software compatible con él ofrece un abanico infinito de posibilidades. Además, al tratarse de software libre, son muchos los fabricantes de componentes compatibles con estos microcontroladores, por lo que el precio de los mismos es generalmente muy reducido y su disponibilidad está asegurada.

2.1.3. Microcontroladores Atmel y placas de desarrollo.

Como se ha indicado anteriormente, los microcontroladores Atmel son compatibles con el IDE de Arduino. Son especialmente interesantes los circuitos integrados sobre placa basados en estos microcontroladores.

Estos circuitos integrados (placas) constan de un microcontrolador Atmel y otros elementos activos y/o pasivos como pines de entrada y salida o conectores que facilitan la conexión de periféricos al microcontrolador; entendiendo como periféricos todo tipo de sensores, shields de comunicaciones y demás componentes.

Entre la gama de circuitos integrados (placas) comerciales destacan los basados en el microcontrolador ATmega328P y en el ATMega2560. A continuación, se resumen las características principales de ambos:

Microcontrolador	ATmega328P	ATmega2560
Voltaje	5 V	
Alimentación (recomendada)	7-12 V	
Pines digitales	14	54
Pines analógicos	6	15
Memoria Flash	32 Kb	256 KB
SRAM	2 KB	8 KB
Frecuencia de muestreo	16 MHz	16 MHz

Tabla 1 Comparación microcontroladores Atmel

Existen diversos fabricantes de circuitos integrados basados en los microcontroladores Atmel, entre ellos destacan marcas como Arduino y Elegoo.

En esta ocasión, se trabajará con un circuito integrado sobre placa basado en el microcontrolador ATmega2560. Se ha escogido este modelo por diversas razones, principalmente, por el gran número de entradas analógicas que ofrece (15) y por su gran capacidad de memoria flash, capaz de almacenar y ejecutar códigos pesados durante periodos indefinidos de tiempo.

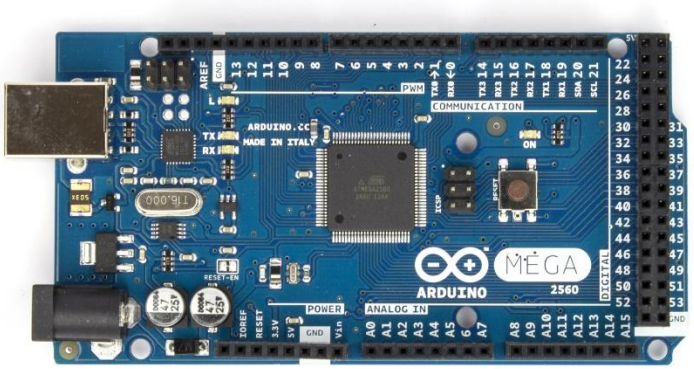


Fig 3 Placa de expansión para AtMega2560

De ahora en adelante, se utilizará “placa” para referirse al circuito integrado basado en el microcontrolador ATmega2560.

2.2. Medida de la corriente AC

2.2.1. Introducción

La medida de corriente alterna es necesaria tanto para cuantificar la generación fotovoltaica como el intercambio de energía con la red eléctrica. Esta medida se debe llevar a cabo necesariamente de forma no invasiva ya que de no ser así obligaría a cortar la corriente de toda la instalación, lo cual a menudo puede resultar inviable cuando se pretende monitorizar una instalación en uso.

2.2.2. Elección del método de sensado

Existen dos maneras de medir la corriente alterna: mediante resistencias shunt o mediante transformadores de corriente, también conocidos como pinzas amperimétricas.

a. Resistencias shunt

La medición de corriente mediante resistencias *shunt* es un método invasivo, ya que requiere introducir una resistencia en serie con el circuito que se quiere medir, lo cual se debe evitar, tal y como se ha comentado anteriormente. Además, este método de sensado produce pérdidas de carga ya que se basa en la ley de Ohm ($V = I * R$) para generar una ligera caída de tensión en bornes de la resistencia de medida, lo cual conlleva una disipación de potencia ($P = I^2 * R$) a tener en cuanta cuando la corriente que la atraviesa es de gran valor.

b. Pinzas amperimétricas. Transformadores de corriente.

Las pinzas amperimétricas son transformadores de corriente no invasivos, ya que son transformadores con núcleo dividido, permitiendo su colocación sin necesidad de desconectar el cable a medir. Además, puesto que la medida se realiza de forma indirecta no producen pérdidas de carga en el circuito principal y ofrecen aislamiento galvánico entre el circuito de señal y el de potencia.

El principio de funcionamiento de los transformadores de corriente es similar a los transformadores de tensión. En esta ocasión, cuentan con un devanado primario (a menudo el propio conductor), un núcleo magnético y un devanado secundario con un mayor número de espiras. La corriente alterna que circula por el conductor (devanado primario) produce un campo magnético variable en el tiempo, el cual induce una corriente alterna en el devanado secundario, cuyo valor depende de la relación entre el número de espiras de ambos devanados.

$$I_1 N_1 = I_2 N_2$$

Al tratarse de una medida indirecta, existe una única espira en el devanado primario ($N_1 = 1$), por tanto:

$$I_2 = \frac{I_1}{N_2}$$

Habitualmente, los sensores de corriente comerciales introducen una resistencia (R) de valor conocido para ofrecer una salida en voltaje, lo cual facilita la lectura de la misma mediante cualquier tipo de convertidor analógico digital.

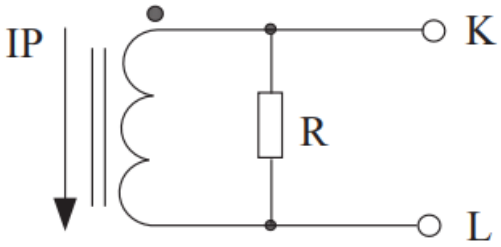


Fig 4 Funcionamiento sensores SCT-013-0xx

Existe gran variedad de sensores de transformadores de corriente no invasivos en el mercado, entre ellos, destaca la gama de sensores SCT-013-0xx, los cuales ofrecen soluciones para medir distintos rangos de tensiones y sensores con salida tanto en corriente como en tensión.

Model	SCT-013-000	SCT-013-005	SCT-013-010	SCT-013-015	SCT-013-020
Input current	0-100A	0-5A	0-10A	0-15A	0-20A
Output type	0-50mA	0-1V	0-1V	0-1V	0-1V
Model	SCT-013-025	SCT-013-030	SCT-013-050	SCT-013-060	SCT-013-000V
Input current	0-25A	0-30A	0-50A	0-60A	0-100A
Output type	0-1V	0-1V	0-1V	0-1V	0-1V

※ Output type: voltage output type built-in sampling resistor, current output type built-in protective diode.

Tabla 2 Gama de sensores SCT-013

Además, los sensores de dicha gama ofrecen una salida de tipo mini Jack stereo, lo que permite diseñar un medidor con pinzas amperimétricas intercambiables, permitiendo captar de forma precisa diferentes rangos de potencia.

Fig 5 Sensor SCT-013-030

2.2.3. Acomodo de la señal de salida del sensor

Se observa que todos los sensores de la gama SCT-013-0xx ofrecen un voltaje a la salida del sensor, excepto el SCT-013-000, que proporciona una salida en corriente. Además, la salida en tensión de todos ellos varía en el mismo rango de tensión (0-1 V rms), lo cual permite realizar un circuito de adecuación de la señal para todas las pinzas, independientemente de la corriente máxima que permiten medir.

2.2.4. Señal de salida de los sensores SCT-013-0xx

Tal y como se ha comentado anteriormente, los sensores con los que se va a trabajar ofrecen una señal de salida en forma de tensión alterna, cuyo valor rms oscilará entre 0 y 1 V, es decir, se tratará de una onda alterna de frecuencia igual a la frecuencia de la corriente medida (habitualmente los 50 Hz de la red), cuyo valor pico será el siguiente:

$$V_{rms} = \frac{V_{max}}{\sqrt{2}}$$

$$V_{max} = \sqrt{2}\, V_{rms} = 1.4142\ V$$

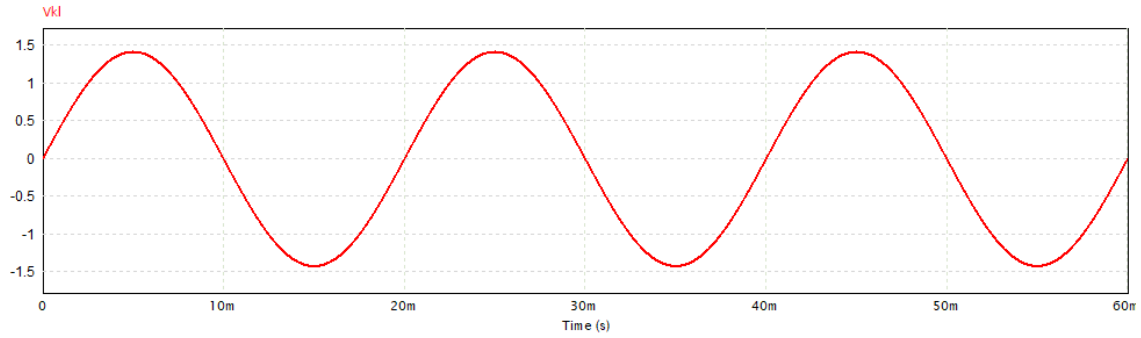


Fig 6 Salida de los sensores SCT-013

Se observa la necesidad de adecuar la señal de salida de los sensores SCT-013-0xx al rango de tensión de entrada del convertidor analógico digital de la placa, el cual es de 0 a 5V.

La adecuación de la señal se lleva a cabo en dos etapas, en primer lugar, se amplificará la salida del sensor, de manera que la amplitud pico-pico de la misma sea de 5V y finalmente se añadirá un offset de 2,5 V (DC) para centrar la señal oscilante en el rango de medida del ADC del circuito integrado.

2.2.5. Circuito de offset

El convertidor analógico digital de la placa no permite tensiones negativas a la entrada, por lo que se añade un offset DC a la señal de salida del sensor para centrar la misma en el rango de entrada del ADC. Para ello, se añade un offset de 2,5V mediante un divisor de tensión, tal y como muestra la figura 7.

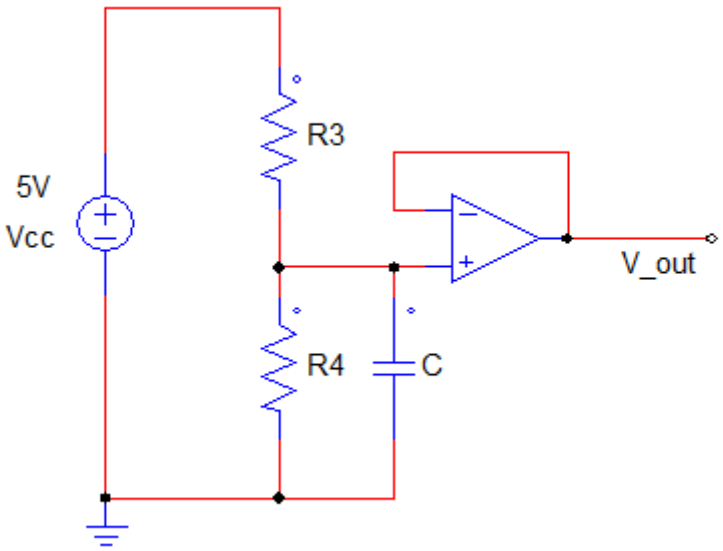


Fig 7 Circuito de offset

El divisor de tensión divide la tensión Vcc de la placa (5V) a la mitad mediante dos resistencias iguales. Para que el consumo de corriente sea mínimo el valor de las resistencias utilizadas es elevado, sin embargo, un valor excesivo podría desestabilizar la tensión en el punto medio. En la práctica, ambas resistencias son de 10 kΩ.

$$R_3 = R_4 = 10\ k\Omega$$

Además, se añade un condensador (C) en paralelo con la resistencia (R4) para estabilizar la salida del divisor de tensión y filtrar cualquier desequilibrio que pudiera producirse. El valor de dicho condensador es de 10 µF.

Por último, se añade un amplificador operacional en modo seguidor de tensión para aislar el circuito de offset del resto del circuito de captación de señal.

Mediante la herramienta de simulación de Psim se comprueba el correcto funcionamiento del divisor de tensión.

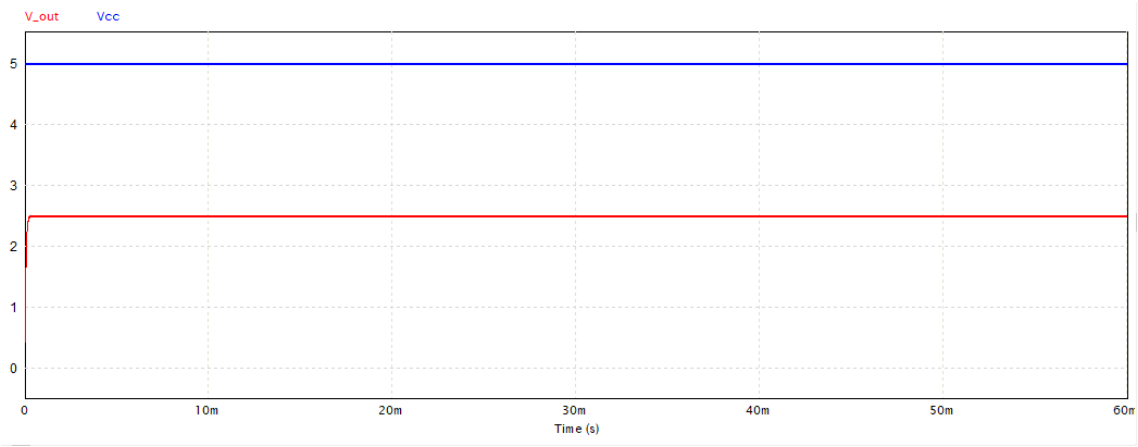


Fig 8 Salida del circuito de offset

Se observa que el condensador produce un retraso de unos pocos µs, lo cual es despreciable para esta aplicación puesto que la conexión se realizará de forma permanente.

2.2.6. Circuito amplificador no inversor

Para aprovechar al máximo el rango de tensión de entrada del ADC de la placa, se debe amplificar la diferencia de tensión que el sensor de corriente genera entre sus dos bornes de salida, denominados K y L por el fabricante (ver figura 4) para que su amplitud pico-pico se corresponda con el span del ADC, 5V. Dicha tensión senoidal, con amplitud pico-pico de 2.82843 V se amplificará para obtener una tensión pico-pico máxima de 5V. La ganancia total del circuito amplificador será la siguiente:

$$G = \frac{5\ V}{2 * \sqrt{2}} = 1.767767$$

Se trata de amplificar una señal diferencial, es decir, la diferencia de tensión existente entre dos puntos del circuito, rechazando el offset DC, el cual es el mismo para los dos puntos.

El diseño y análisis del circuito de amplificación diferencial no inversor se basa en las propiedades de los amplificadores operacionales ideales. Se debe tener en cuenta que los AO ofrecen una gran impedancia de entrada, por lo que su absorción de corriente es prácticamente nula y despreciable.

El circuito de amplificación diferencial con ganancia controlada diseñado es el siguiente:

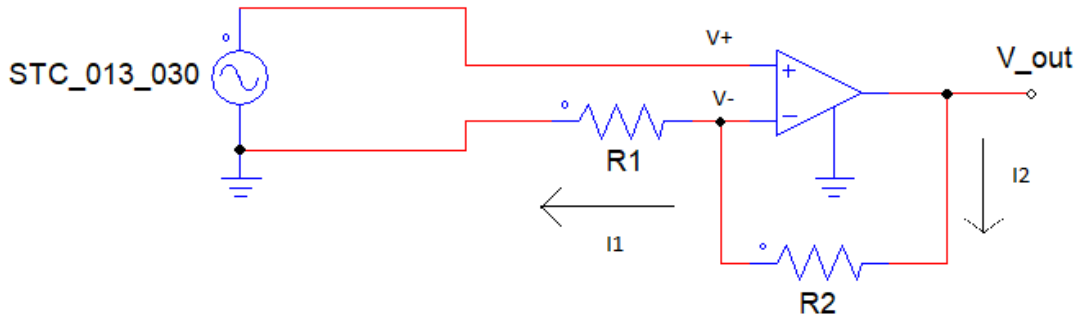


Fig 9 Circuito amplificador no inversor

Para el análisis de circuitos con amplificadores operacionales se deben tener en cuenta dos consideraciones:

1. La resistencia de entrada es mayor que cualquier otra resistencia del circuito. La absorción de corriente es nula.
2. Cuando se opera en modo lineal (no saturado) la diferencia de tensión entre las dos entradas es despreciable, por tanto:

$$V_+ \cong V_-$$

De esta manera, el comportamiento del circuito diseñado es el siguiente, tomando como referencia de tierra el borne L de la salida del sensor.

$$V_+ \cong V_- = V_{in}$$

$$I_1 = I_2 = I$$

$$I = \frac{V_{in}}{R_1}$$

$$V_{out} = V_{in} + I * R_2 = V_{in}\left(1 + \frac{R_2}{R_1}\right)$$

Se observa que la ganancia de este circuito es siempre mayor que la unidad y que se puede controlar su valor modificando el valor de las resistencias.

$$G = 1 + \frac{R_2}{R_1} = 1.767767$$

$$R_1 = 1.3025\, R_2$$

Finalmente, se comprueba el correcto funcionamiento del circuito amplificador diferencial gracias a la simulación realizada en Psim. De nuevo, se toma como tierra para la simulación el borne L del sensor.

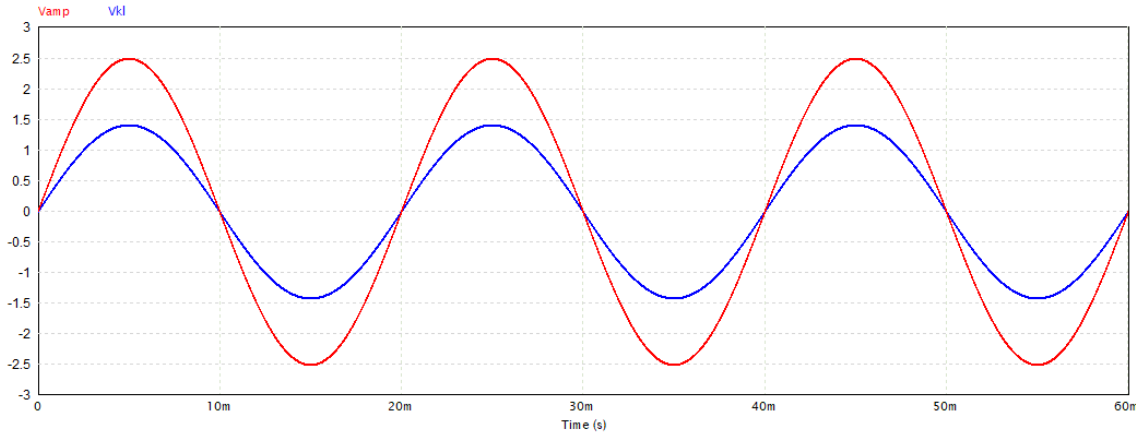


Fig 10 Salida del circuito amplificador no inversor

Se comprueba que se obtiene una señal alterna, en fase con la de entrada con amplitud pico-pico de 5V.

Con el fin de conseguir la ganancia más cercana posible a la calculada anteriormente se escogen los siguientes valores normalizados de resistencias:

$$R_1 = 4.7\ k\Omega$$

$$R_2 = 3.9\ k\Omega$$

2.2.7. Adecuación de la señal de salida de los sensores SCT-013-0xx.

Finalmente, se implementa conjuntamente el circuito de offset DC y el amplificador diferencial con la ganancia adecuada. Es importante no referenciar a tierra la salida del sensor para que el offset sea efectivo. La unión de ambos circuitos se realizará a la salida del sensor, ya que el circuito de amplificación trabaja en modo diferencial, por lo que no amplificará en ningún caso el componente DC de la señal.

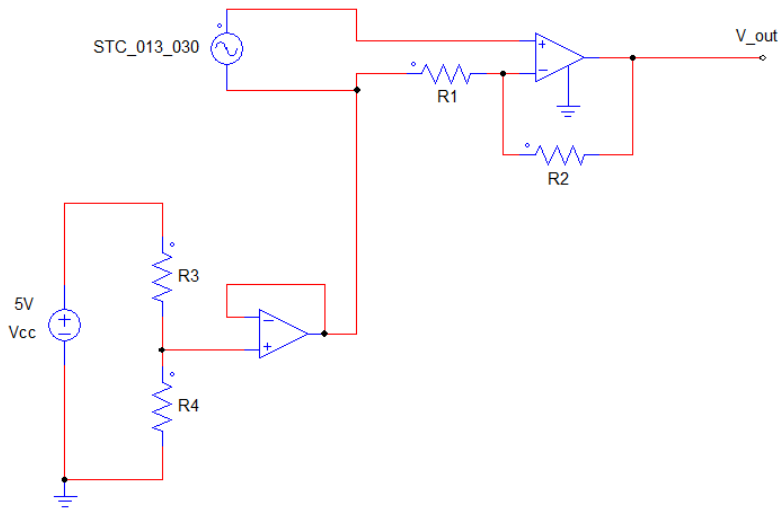


Fig 11 Circuito de adecuación de la señal para los sensores SCT-013-0xx

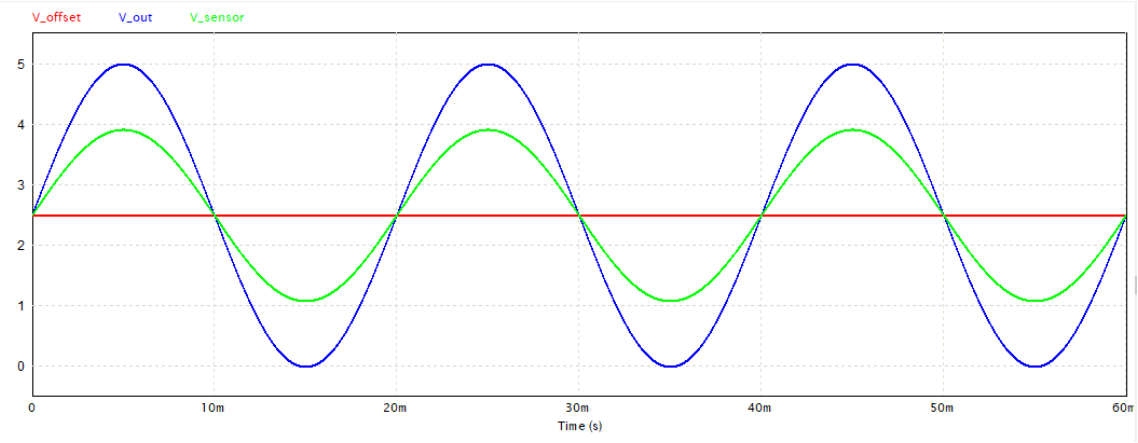


Fig 12 Salida acomodada de los sensores SCT-013

Se observa como la señal de entrada al ADC de la placa oscila ahora entre 0 y 5V cuando la corriente que circula por el conductor es la máxima que mide el sensor SCT-013-0xx, independientemente del modelo escogido, siempre que sea uno de los que ofrece salida en tensión (ver tabla 2).

2.2.8. Programación de la lectura del sensor de corriente AC

a. Lectura de la corriente AC. Librería EmonLib.

El código de Arduino para la lectura de los sensores de corriente alterna SCT-013-0xx se basa en la librería Emonlib creada por el proyecto Open Energy Monitor. Esta librería es de código abierto y compatible con el IDE de Arduino.

En primer lugar, se instala la librería en el IDE de Arduino. Para ello, se busca "emonlib" en el "Gestor de Librerías" del IDE de Arduino y se instala la librería ***EmonLib***, creada por *OpenEnergyMonitor*.

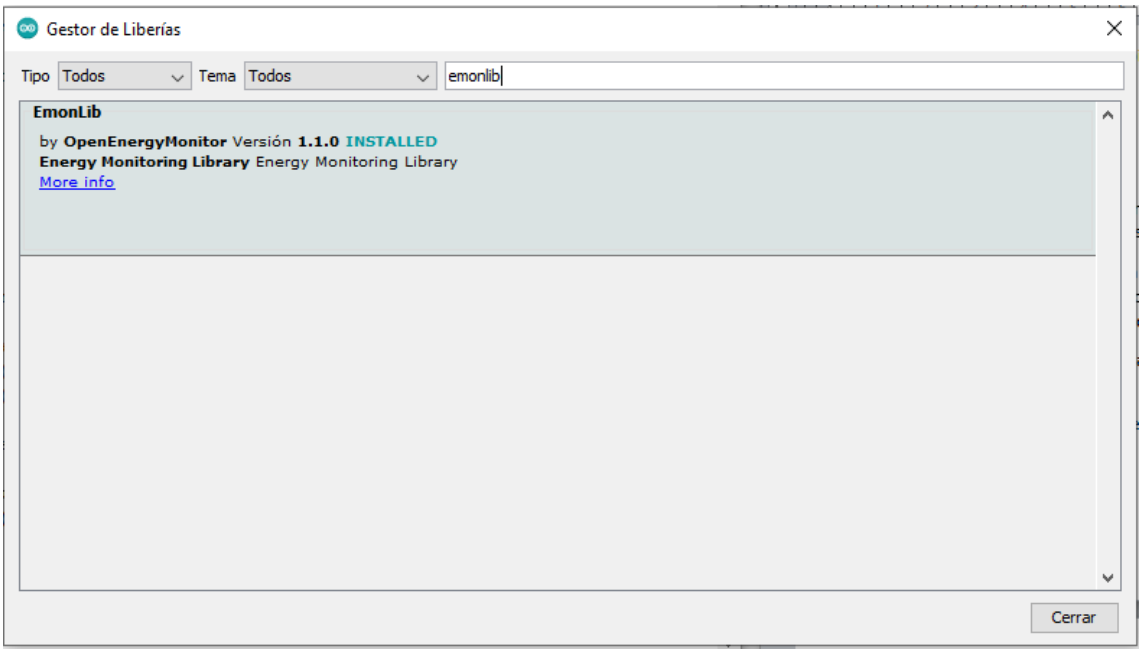


Fig 13 Gestor de librerías

Como casi todas las librerías que se encuentran en el IDE de Arduino, la librería EmonLib ofrece ejemplos de código para dar a conocer la sintaxis necesaria para su utilización.

Para la lectura de la corriente AC se utiliza como base el ejemplo "Current Only" de la librería EmonLib, el cual se muestra a continuación:

current_only §

```
// EmonLibrary examples openenergymonitor.org, Licence GNU GPL V3

#include "EmonLib.h"                   // Include Emon Library
EnergyMonitor emon1;                   // Create an instance

void setup()
{
  Serial.begin(9600);

  emon1.current(1, 111.1);             // Current: input pin, calibration.
}

void loop()
{
  double Irms = emon1.calcIrms(1480);  // Calculate Irms only

  Serial.print(Irms*230.0);         // Apparent power
  Serial.print(" ");
  Serial.println(Irms);          // Irms
}
```

b. Calibración de los sensores SCT-013-0xx

Para que la medida de corriente sea fiel a la realidad se debe calibrar correctamente el sensor, ya que existen diversos factores que pueden hacer que esta no sea necesariamente precisa. Entre estos factores a tener en cuenta destacan la dispersión que puede existir en el valor de la resistencia de carga del sensor y la precisión del ADC a la hora de medir el voltaje salida del sensor.

El proceso de calibración del sensor se lleva a acabo de forma experimental, comparando la medida obtenida en tiempo real mediante el sensor y la de un aparato de medida de referencia.

En esta ocasión se utilizará un vatímetro digital que incorpora una pinza amerimétrica como aparato de medida de referencia, el cual utiliza un sensor de efecto Hall para medir corriente AC y/o DC de forma no invasiva.

Fig 14 Pinza ampermétrica comercial

La calibración experimental que se realizar experimentalmente sirve para un sensor SCT-013-030 en particular, este proceso deberá repetirse con cada uno de los sensores de esta gama que se utilicen, para realizar las mediciones de la forma más precisa posible.

El factor de calibración que se debe ajustar aparece indicado en rojo a continuación:

```
emon1.current(1, 111.1);             // Current: input pin, calibration.
```

Para el proceso de calibración se mide la corriente que circula por el cable de alimentación de un cargador de baterías de 1200 W. Este cargador se alimenta con la red doméstica de baja tensión (230V) por lo que cuando trabaje a potencia nominal, cargando las baterías en modo *bulk,* circulará por la alimentación una corriente cercana a los 5,2 A.

La medida obtenida con la pinza amperimétrica comercial ronda los 3.75 A (ver figura X).

Fig 15 Medida de la corriente

Al mismo tiempo, se miden los resultados obtenidos mediante la programación antes comentada:

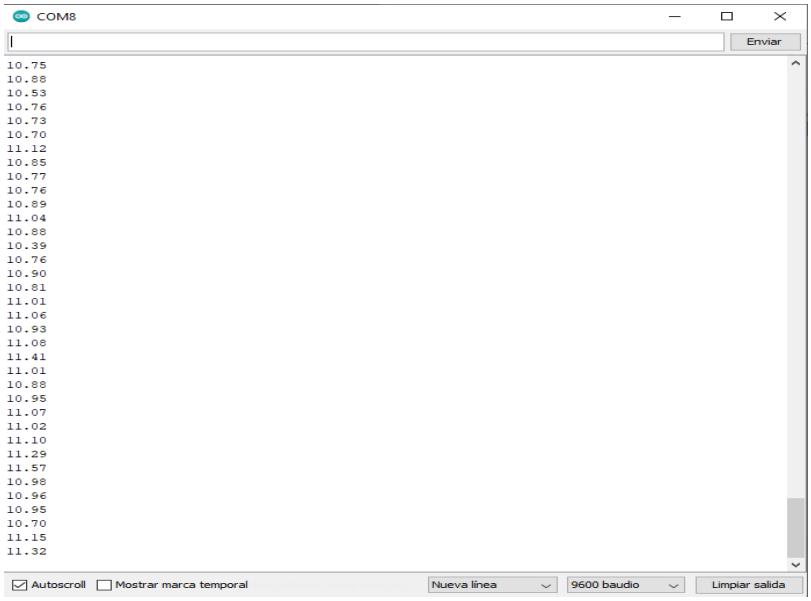


Fig 16 Calibración del sensor

Se observa que ambos resultados difieren notablemente y se procede a cambiar el parámetro de la calibración del sensor hasta que la medida que proporciona el Arduino se corresponda con la realidad.

El valor del parámetro de calibración finalmente se debe ajustar en 35 para la pinza de 30 A SCT-013-030. Los resultados obtenidos una vez calibrado el sensor son los siguientes:

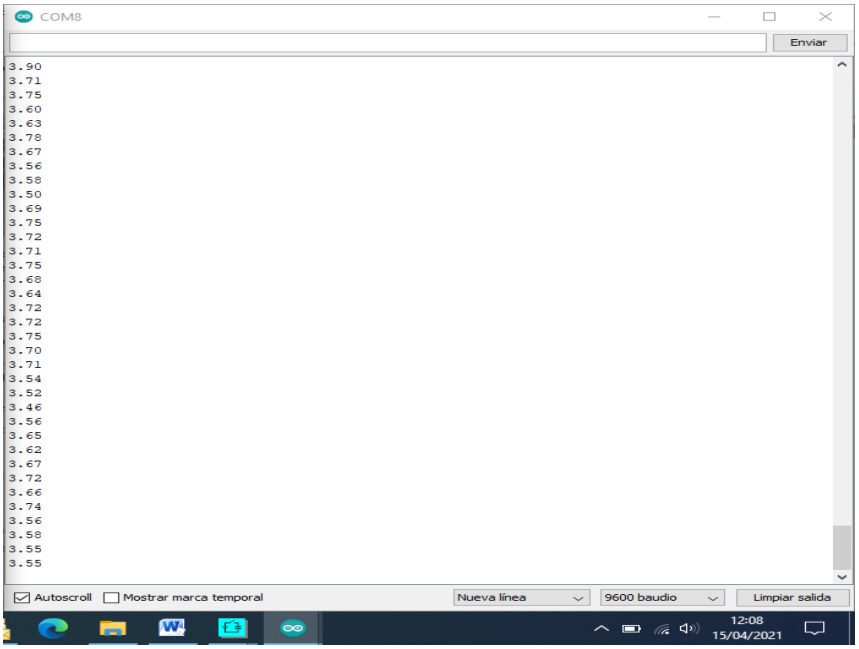


Fig 17 Resultados de la calibración

Cabe destacar que el parámetro de calibración depende del modelo de sensor que se utiliza (SCT-013-030 p.ej.) y se comprueba que dicho valor es el mismo para todos los sensores del mismo tipo, por lo que el proceso de calibración se repetirá cuando se cambie el sensor por uno con capacidad de medida diferente y no cuando se intercambie el sensor por una de la misma clase.

c. Programación final para la lectura de corriente alterna.

Finalmente, el script de programación utilizado para la lectura de la corriente alterna (con la pinza de 30A) es el siguiente:

Corriente_AC

```
//LIBRERÍA PARA CALCULAR LA CORRIENTE RMS
#include "EmonLib.h"

EnergyMonitor emon1;                    //Se crea una instancia para la librería

void setup()
{
  Serial.begin(9600);

//IMPORTANTE: Calibrar correctamente el sensor
//CALIBRACIÓN (experimental): 35 para la pinza de 30A

  emon1.current(3, 35);                 // Current: input pin, calibration.

}

void loop()
{
  double Irms = emon1.calcIrms(1480);   // Cálculo de corriente rms
  Serial.println(Irms);                 // Irms
  delay(500);
}
```

2.3. Medida de la tensión AC

2.3.1. Introducción

La medida de la tensión alterna es necesaria tanto para la cuantificación de la generación fotovoltaica como para la conocer el intercambio de potencia con la red eléctrica. Además, en sistemas trifásicos, la tensión de cada una de las fases puede servir como indicador de desequilibrios, igual que la medida de la corriente.

Se deben extremar las precauciones a la hora de medir la tensión alterna dada su magnitud típica, ya que las tensiones típicas con las que se trabajará serán 400/230 V.

La tensión típica en sistemas monofásicos es de 230 V rms de línea. En sistemas trifásicos las tensiones típicas son de 400 V de línea y 230 V de fase. Se decide medir siempre la tensión de fase ya que de esta manera se reduce el riesgo de accidente al manejar tensiones más bajas.

2.3.2. Elección del método de sensado

Existen dos maneras ampliamente extendidas para medir la tensión en un circuito eléctrico, ya se AC o DC. A continuación, se presentan ambos métodos y se escoge el método de sensado.

a. Divisor de tensión

La primera manera de medir la tensión en un circuito eléctrico que se plantea es la utilización de un divisor de tensión resistivo. Se trata de un circuito compuesto por dos resistencias en serie, las cuales por la ley de Ohm reducen la tensión de entrada, cuando esta es excesivamente alta para ser medida directamente, ofreciendo una tensión manejable para el ADC de la placa (inferior a 5 V).

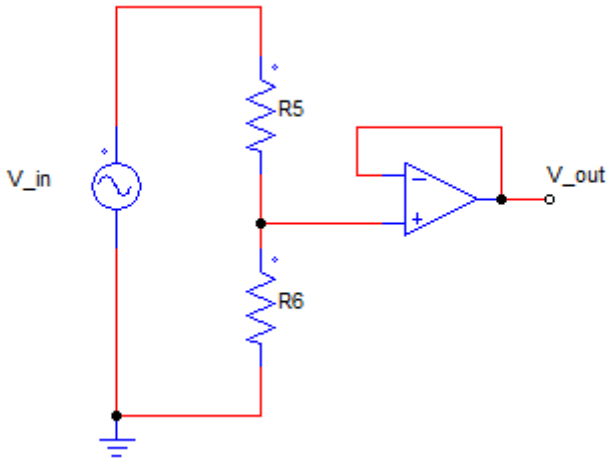


Fig 18 Divisor de tensión

Este es un método de sensado sencillo y barato, sin embargo, presenta algún impedimento cuando la tensión a la entrada es alta y la reducción del divisor de tensión debe ser grande. En primer lugar, la relación entre las tensiones de entrada y salida exige una gran diferencia entre los valores de las resistencias del divisor de tensión. Para un sistema monofásico de 230V de fase:

$$\frac{V_{in}}{V_{out}} = \frac{230\sqrt{2}}{5} = 65$$

Teniendo esta relación entre ambas tensiones, la relación entre las resistencias del divisor de tensión debe ser la siguiente:

$$R_1 = 64\, R_2$$

Ante esto, se observa que cualquier desviación del valor de las resistencias respecto a su valor nominal, lo cual es habitual, supondrá una gran pérdida de precisión. Por tanto, los divisores de tensión resultan poco precisos para medir grandes tensiones.

Los divisores de tensión, además, no ofrecen aislamiento galvánico entre el circuito de potencia y el de señal, pese a que el AO colocado a la salida del mismo (ver figura 18) proporcione cierto nivel de aislamiento y protección. Por tanto, resultan aplicaciones algo más peligrosas que los transformadores de corriente.

b. Transformadores de tensión

Los transformadores de medida se basan en el principio de funcionamiento de los transformadores para reducir altas tensiones y/o corrientes hasta niveles de tensión (o corriente) medibles por los aparatos de medida.

El devanado primario del transformador de tensión se conecta en paralelo con el circuito de potencia y este genera un nivel de tensión mucho más reducido en el devanado secundario, dependiendo de la relación de transformación del transformador.

Existen transformadores de medida diseñados específicamente para controladores como Arduino, entre ellos, destaca el transformador de voltaje de módulo activo ZMPT101B. Estos sensores permiten medir corrientes alternas de hasta 250 V rms y ofrece una señal alterna de 5V max a la salida, adecuada al rango de entrada del ADC de los controladores. Existe gran variedad de fabricantes de este tipo de sensores y se encuentran fácilmente por unos 6-7€ (precio de abril de 2021). Además, existen ejemplos de código compatibles con el IDE de Arduino preparados para leer la salida de este tipo de sensores. Por tanto, se escoge el transformador de voltaje de módulo activo ZMPT101B para la medida de corriente alterna.

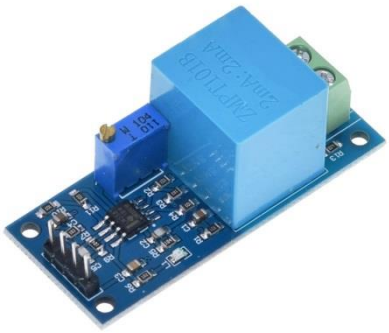


Fig 19 Sensor de tensión AC

Mediante estos sensores se miden las tensiones de fase, tanto en sistemas monofásicos como trifásicos. En el caso de las instalaciones trifásicas el hecho de medir tensiones de fase ofrece dos grandes ventajas, en primer lugar, las tensiones a medir son menores que las de línea y el riesgo que esto supone es menor y en segundo lugar, permite detectar cualquier desequilibrio que pueda producirse entre las fases.

2.3.3. Conexión con la placa

La conexión del sensor con la placa es sencilla y se realiza de la siguiente manera:

a. Patilla ACC del sensor al pin 5V de la placa
b. Patilla OUT del sensor al pin analógico de entrada de la placa, por ejemplo, el pin A0.
c. Patilla GND del sensor al pin GND de la placa.
d. Una patilla GND del sensor queda sin conectar.

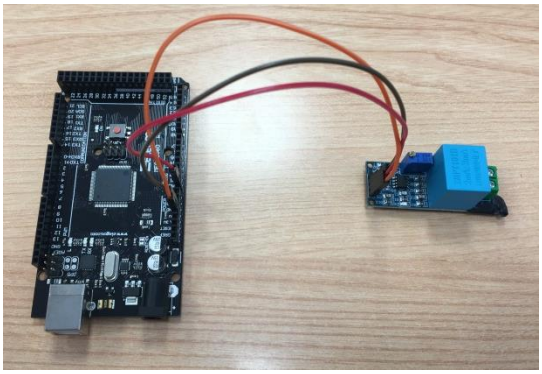


Fig 20 Conexión con la placa: sensor de tensión AC

IMPORTANTE: Aislar las patillas salientes en la parte inferior del sensor para evitar descargas eléctricas por contacto.

2.3.4. Ajuste de la señal de salida del sensor y calibración.

Tal y como se ha comentado anteriormente, los transformadores ZMPT101B ofrecen una salida adecuada para ser medida directamente con Arduino o cualquier placa similar. Sin embargo, la amplitud de la señal de salida debe ajustarse haciendo uso del potenciómetro incorporado en el propio sensor, tal y como se indica a continuación.

El ajuste de la señal de salida del sensor se lleva a cabo gracias a los códigos e instrucciones que ofrece Naylamp Mechatronics.

En primer lugar, se carga el siguiente código en la placa y se ejecuta.

volt_ac_cal §

```
int adc_max=0;
int adc_min=1023;
long tiempo_init;

void setup() {
  Serial.begin(115200);
  tiempo_init=millis();
}

void loop() {

  if( (millis() - tiempo_init) > 500){
    adc_max=0;
    adc_min=1023;
    tiempo_init=millis();
  }

  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);

  if(sensorValue > adc_max){
    adc_max=sensorValue;
  }
  else if(sensorValue < adc_min){
    adc_min=sensorValue;
  }

  Serial.print("adc_max: ");
  Serial.print(adc_max);
  Serial.print("   adc_min: ");
  Serial.println(adc_min);

  delay(1);        // delay in between reads for stability
}
```

Este código calcula los valores mínimos y máximos medidos por el ADC de la placa y los almacena en las variables “adc_max” y “adc_min”. Con el código cargado se ajusta el valor del potenciómetro hasta que el valor de “adc_max” sea 700 bit.

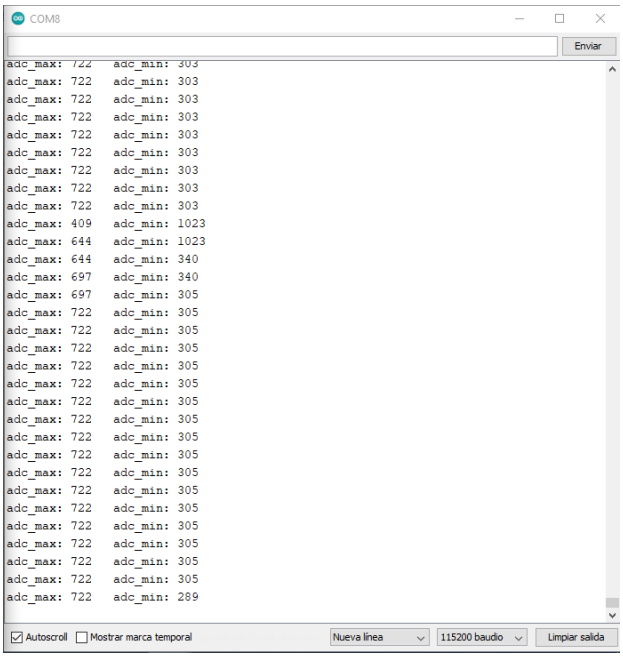


Fig 21 Calibración del sensor de tensión AC

Se observa que la medida es inestable cuando se conecta la salida del sensor directamente a la placa y se decide añadir un filtro paso bajo para filtrar el ruido electromagnético que pueda estar afectando a la medida. La frecuencia de corte del filtro deberá ser mayor que la frecuencia fundamental de la onda que se pretende medir, los 50 Hz (60 Hz en algunos países) de la red eléctrica concretamente.

Se utilizará un filtro paso bajo de primer orden de tipo RC, cuyas características serán las siguientes:

$$\boldsymbol{f_{corte} \cong 100\ Hz}$$

Se decide utilizar un condensador de 1 µF, siendo este un valor normalizado y fácil de conseguir en el mercado.

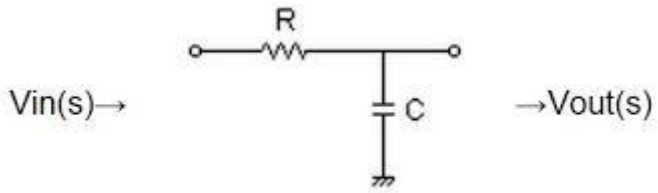


Fig 22 Filtro Paso Bajo de primer orden

$$f_{corte} = \frac{1}{2\pi RC} = 100\ Hz$$

$$R = \frac{1}{2\pi C f_{corte}} = 1591.55\ \Omega$$

Se utilizará la resistencia comercial con el valor más cercano posible, en este caso, la resistencia utilizada será de 1,5 kΩ. Por tanto, el filtro de paso bajo utilizado tendrá las siguientes características:

$$C = 1\ \mu F$$

$$R = 1{,}5\ k\Omega$$

$$f_{corte} = \frac{1}{2\pi RC} = 106.1\ Hz$$

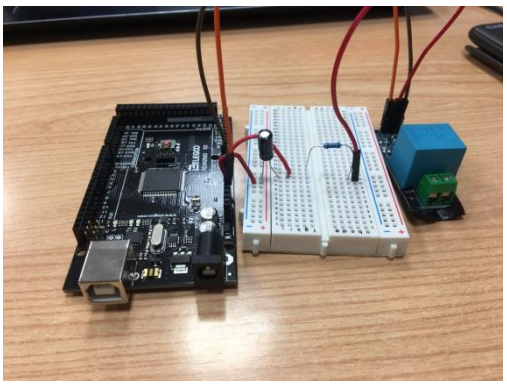


Fig 23 Circuito de sensado y filtro

Se realiza de nuevo la toma de datos y se observa como las medidas se estabilizan notablemente.

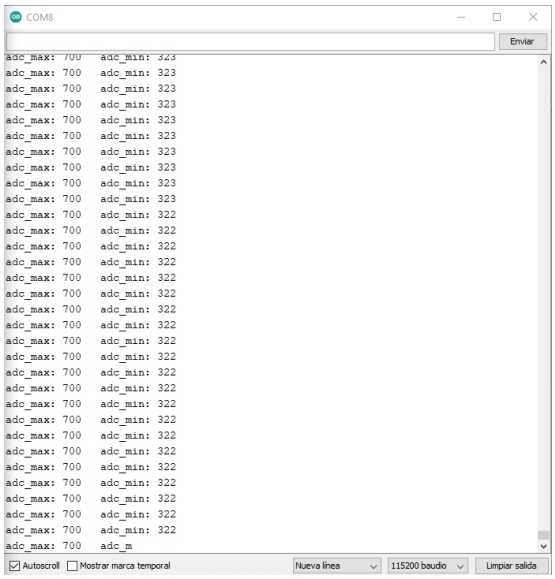


Fig 24 Resultados tras filtrado

Cuando se ajusta el valor del potenciómetro correctamente se deben registrar los valores de "adc_max", "adc_min" y medir con un multímetro el valor de la tensión real de la red en ese instante. Estos valores se utilizan como referencia para la lectura del sensor.

Los valores obtenidos en el proceso de calibración son los siguientes:

e. adc_max = 700.
f. adc_min = 322.
g. V_red = 230.1 V

Estos valores se introducen en el siguiente script de Arduino para completar la programación del sensor de voltaje AC.

volt_ac_inst

```
int adc_max=700;        //Reemplazar por valor adc_max entregado por el sketch: volt_ac_cal
int adc_min=322;        //Reemplazar por valor adc_min entregado por el sketch: volt_ac_cal
float volt_multi=230.1;   //Reemplazar por el "voltaje ac rms" entregado por un multimetro
float volt_multi_p;
float volt_multi_n;

void setup() {

  Serial.begin(115200);
  volt_multi_p = volt_multi * 1.4142;   //Voltaje pico= Voltaje RMS * 1.4142 (Corriente Monofasica)
  volt_multi_n = volt_multi_p * -1;
}

void loop() {

  float volt_rms=get_voltage(); //Voltage eficaz (V-RMS)


  Serial.print("Vrms: ");
  Serial.print(volt_rms,3);
  Serial.println(" VAC");
  //delay(100);
}
```

volt_ac_inst

```
float get_voltage(void)
{
  float adc_sample;
  float volt_inst=0;
  float Sumatoria=0;
  float volt;
  long tiempo_init=millis();
  int N=0;

  while( (millis() - tiempo_init) < 600)//Duración 0.6 segundos(Aprox. 30 ciclos de 50Hz)
  {
    //adc_sample = analogRead(A0) - 510;////voltaje del sensor
    //volt_inst = map(adc_sample,-188,188,-310,310);

    adc_sample = analogRead(A0);    //voltaje del sensor
    volt_inst = map(adc_sample,adc_min,adc_max,volt_multi_n,volt_multi_p);
    Sumatoria = Sumatoria+sq(volt_inst);    //Sumatoria de Cuadrados
    N = N+1;
    delay(1);
  }

  //Serial.print("N: ");
  //Serial.println(N);

  volt=sqrt((Sumatoria)/N); //ecuación del RMS
  return(volt);
}
```

Este código muestrea la lectura del sensor durante 600 ms (30 ciclos de 50 Hz) y calcula el valor ***true rms*** de la tensión medida por el sensor. Para ello, calcula el sumatorio de cuadrados de los valores obtenidos, los divide entre el número de medidas y calcula la raíz cuadrada del resultado.

$$V_{trms} = \sqrt{\frac{{V_1}^2 + {V_2}^2 + \cdots + {V_n}^2}{n}}$$

Este código se debe ajustar para tomar tantas medidas de voltaje como sean necesarias, dependiendo de si se está monitorizando una instalación monofásica o trifásica.

El bucle get_voltaje() deberá devolver dos medidas de voltaje en las instalaciones monofásicas y seis en la trifásicas. Para tomar los valores de voltaje de forma simultánea se deberá ajustar el código de la siguiente manera. Se muestra el código para la medida de 6 voltaje de fase en una instalación trifásica:

En primer lugar, desde el código principal se realiza una llamada al bucle get_voltage() para que devuelva las seis medidas de tensión de fase:

```
float Vinv1, Vinv2, Vinv3, Vred1, Vred2, Vred3 =get_voltage(); //Voltage eficaz (V-RMS)
```

Una vez inicializado el bucle, se declaran las variables necesarias para la toma de datos:

```
//CÁLCULO DEL VOLTAJE AC RMS
//IMPORTANTE: NECESITA 5V COMO REFERENCIA ANALÓGICA
//DEVUELVE EL VOLTAJE DE SALIDA DEL INVERSOR Y EL VOLTAJE DE SALIDA DE LA ACOMETIDA
float get_voltage(void)
{
  //analogReference(DEFAULT); //POR SI SE HA CAMBIADO EN OTRO MOMENTO
  float adc_inv1, adc_inv2, adc_inv3, adc_red1, adc_red2, adc_red3;
  float volt_inst1=0;
  float volt_inst2=0;
  float volt_inst3=0;
  float volt_inst_red1=0;
  float volt_inst_red2=0;
  float volt_inst_red3=0;
  float Sum1=0;
  float Sum2=0;
  float Sum3=0;
  float Sum4=0;
  float Sum5=0;
  float Sum6=0;
  float V_inv1, V_inv2, V_inv3, V_red1, V_red2, V_red3;
  long tiempo_init=millis();
  int N=0;
```

A continuación, se realiza la medición y se devuelven las medidas obtenidas:

```
  while( (millis() - tiempo_init) < 600)//Duración 0.6 segundos(Aprox. 30 ciclos de 50Hz)
  {

    adc_inv1 = analogRead(A8);    //Entrada sensor de voltaje del inversor
    adc_inv2 = analogRead(A9);    //Entrada sensor de voltaje del inversor
    adc_inv3 = analogRead(A10);    //Entrada sensor de voltaje del inversor

    adc_red1 = analogRead(A11);    //Entrada sensor de voltaje de la red
    adc_red2 = analogRead(A12);    //Entrada sensor de voltaje de la red
    adc_red3 = analogRead(A13);    //Entrada sensor de voltaje de la red

    volt_inst1 = map(adc_inv1,adc_min_inv1,adc_max_inv1,volt_multi_n_inv1,volt_multi_p_inv1);
    volt_inst2 = map(adc_inv2,adc_min_inv2,adc_max_inv2,volt_multi_n_inv2,volt_multi_p_inv2);
    volt_inst3 = map(adc_inv3,adc_min_inv3,adc_max_inv3,volt_multi_n_inv3,volt_multi_p_inv3);

    volt_inst_red1 = map(adc_red1,adc_min_red1,adc_max_red1,volt_multi_n_red1,volt_multi_p_red1);
    volt_inst_red2 = map(adc_red2,adc_min_red2,adc_max_red2,volt_multi_n_red2,volt_multi_p_red2);
    volt_inst_red3 = map(adc_red3,adc_min_red3,adc_max_red3,volt_multi_n_red3,volt_multi_p_red3);

//Sumatoria de Cuadrados
    Sum1 = Sum1+sq(volt_inst1);
    Sum2 = Sum2+sq(volt_inst2);
    Sum3 = Sum3+sq(volt_inst3);

    Sum4 = Sum4+sq(volt_inst_red1);
    Sum5 = Sum5+sq(volt_inst_red2);
    Sum6 = Sum6+sq(volt_inst_red3);

    N = N+1;
    delay(1);
  }
//ecuación del RMS
  V_inv1=sqrt((Sum1)/N);
  V_inv2=sqrt((Sum2)/N);
  V_inv3=sqrt((Sum3)/N);
  V_red1=sqrt((Sum4)/N);
  V_red2=sqrt((Sum5)/N);
  V_red3=sqrt((Sum6)/N);

  return V_inv1, V_inv2, V_inv3, V_red1, V_red2, V_red3;
}
```

2.4. Medida de la corriente DC

2.4.1. Introducción

La lectura de la corriente continua es una parte fundamental para la monitorización de las baterías en cualquier sistema fotovoltaico con almacenamiento eléctrico, ya sea un sistema aislado o un sistema de autoconsumo con baterías. Además, la lectura de la corriente que entra

o sale del sistema de acumulación es fundamental para cuantificar la energía almacenada y/o cedida por el sistema de almacenamiento.

Además, la monitorización de los parámetros de las baterías permite implementar diversos sistemas de seguridad como protecciones ante sobretensiones, sobredescargas, cortocircuitos o sobrecalentamientos.

La medida de la corriente continua que entra o sale del banco de baterías de una instalación fotovoltaica permite, además de cuantificar el consumo y la generación fotovoltaica, conocer el estado de carga de las baterías y detectar posibles cortocircuitos.

2.4.2. Elección del método de sensado

Existen dos maneras de medir la corriente continua que circula por un conductor: utilizando resistencias *shunt* o mediante sensores de efecto Hall.

a. Sensores de efecto Hall

Los sensores de efecto Hall son sensores magnéticos que detectan y cuantifican el campo magnético que genera la corriente eléctrica circulando por un conductor.

Toda corriente que circula a través de un conductor genera un campo magnético en el plano perpendicular a la corriente, la corriente continua genera un campo magnético constante y la corriente alterna un campo magnético variable en el tiempo. Cuando dicho campo magnético atraviesa perpendicularmente un sensor Hall por el que circula una corriente conocida, se genera un voltaje en bornes del sensor, proporcional a la fuerza del campo magnético generado por el conductor. El voltaje generado es proporcional a la corriente que circula por el conductor, lo cual permite cuantificar dicho valor.

Estos sensores requieren una fuente de alimentación externa para generar la corriente que circula por el sensor, además de un material magnético, lo cual encarece el precio de estos dispositivos.

b. Resistencia Shunt

El uso de resistencias shunt para medir la corriente es un método eficaz y barato, con el inconveniente de ser una técnica invasiva. En el caso de la medida de la corriente que circula a través del cableado de las baterías puede implementarse una técnica invasiva, aunque se pretenda colocar en una instalación en funcionamiento, ya que estos circuitos a menudo implementan un dispositivo de corte (desconectadores o seccionadores) para realizar labores de mantenimiento.

Este tipo de sensores se basan en la ley de Ohm ($V = I * R$), midiendo la caída de tensión que se produce cuando la corriente a medir atraviesa la resistencia de medida (shunt). El valor de las resistencias de medida es muy reducido, de manera que las pérdidas que se producen en ella son mínimas y despreciables.

Dicho esto, la corriente continua que circula por el cableado de las baterías se cuantificará utilizando una resistencia *shunt* comercial.

2.4.3. Adecuación de la señal de salida del sensor

Como se ha indicado anteriormente, cuando la corriente circula a través de la resistencia de medida produce una caída de tensión en bornes de la misma. La diferencia de tensión entre

ambos bornes de la resistencia shunt se calcula como la diferencia de las tensiones absolutas de ambos puntos respecto a un punto común, concretamente la referencia de tierra de placa (GND).

Una vez conocida la diferencia de tensión entre ambos bornes de la resistencia se calculará la corriente que circula por ella de la siguiente manera:

$$V = I * R$$

$$I = \frac{V}{R}$$

Por lo que respecta al aparato de medida, ambas medidas se realizan de forma independiente. Eso sí, ambos circuitos de medida son idénticos.

El valor de las resistencias shunt comerciales suele ser tal que cuando circula por ellas la corriente máxima (100, 200, 300 A, depende del modelo) se produzca una caída de tensión de no más de 75 mV. Por tanto, las señales a medir serán dos voltajes de unos pocos mV, es decir, poco energéticas. Esto hace que el sistema de captación de las señales sea especialmente sensible ante cualquier tipo de ruido o interferencia electromagnética (EMI), por lo que se deberá implementar un filtro paso bajo (LPF) en cada uno de los circuitos de medida.

a. Filtro paso bajo (LPF)

El filtro implementado es el mismo para cada uno de los circuitos de captación del voltaje de cada borne de la resistencia.

Se trata de un voltaje continuo, de frecuencia nula, cuyas variaciones son lentas. Por tanto, la frecuencia de corte del filtro puede ser muy pequeña, asegurando un filtrado efectivo de las interferencias electromagnéticas.

Tras comprobar experimentalmente que la frecuencia de corte debe ser cercana a los 0 Hz, se decide implementar un filtro con las siguientes características.

$$R_f = 4.7\ k\Omega$$

$$C_F = 10\ \mu F$$

$$f_{corte} = \frac{1}{2\pi RC} = 3.386\ Hz$$

Se comprueba como el filtro funciona correctamente de manera teórica y experimental. A continuación, se muestra una simulación en PSIM del sistema de medida:

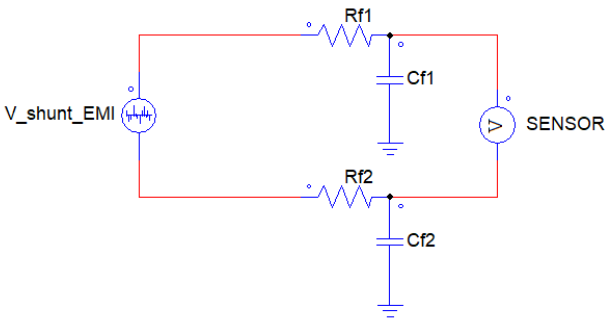


Fig 25 Filtrado de la señal de salida del shunt

La fuente de tensión "V_shunt_EMI" simula la caída de tensión en una resistencia shunt, en este caso se trata de señal de 75 mV de valor medio con un ruido electromagnético de alta frecuencia superpuesto. Esta señal se filtra en ambos extremos de la resistencia shunt y se mide con el sensor para comprobar la efectividad del filtrado.

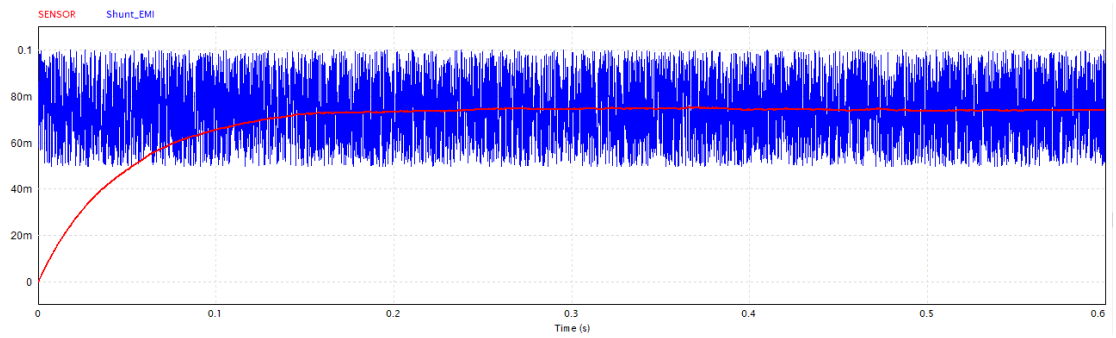


Fig 26 Simulación EMI y filtrado

Se observa como la medida del sensor (rojo) es únicamente el valor medio de la caída de tensión en la resistencia shunt, por lo que se puede afirmar que el filtrado es correcto. Además, cabe destacar que el tiempo de respuesta del filtro es de unos 10 ms con los condensadores inicialmente descargados, lo cual es totalmente despreciable para este uso, ya que las variaciones serán mucho más lentas (se ha simulado un pico de corriente de 0 a la corriente máxima que permite medir el shunt) y dicho retardo no compromete la medida en ningún caso.

2.4.4. Conexión con la placa

El montaje del circuito de medida es sencillo, tal y como se muestra en la figura 25. La medida de las tensiones de ambos puntos se realizará con el convertidor analógico de la placa MEGA2560, por lo que la salida de cada uno de los filtros, que estarán referenciados a la tierra de la placa, se conectará directamente a una de las 16 entradas analógicas de la misma.

La resistencia shunt deberá conectarse en uno de los cables que conectan la batería con el regulador de carga y/o el inversor, dependiendo de la topología de la instalación. Es importante que siempre se conecte en el mismo polo de la batería, para que el criterio de signos utilizado en la medida sea siempre válido.

Se decide colocar la resistencia shunt en serie con el conductor que sale del polo negativo de la batería. La medida de la tensión del borne de la resistencia más cercano a la batería se realizará con el pin A0 de la placa y la medida del borne más alejado de esta con el pin A1. De igual manera, es vital respetar estas conexiones para poder aplicar un criterio de signos común para todas las instalaciones.

2.4.5. Programación

En este apartado se muestra cómo se cuantifica por medio del procesador MEGA2560 la corriente que circula por el cableado de las baterías de la instalación. Además, se debe indicar el sentido de esta corriente con el fin de conocer la dirección del flujo de potencia.

En primer lugar, se mide la diferencia de tensión entre ambos bornes de la resistencia shunt. Como se ha indicado anteriormente, se trata de dos señales de unos pocos mV, por lo que en caso de utilizar la referencia que el convertidor ADC de la placa tiene por defecto, 5V, la resolución sería mala, concretamente:

$$resolución = \frac{V_{ref}}{2^N} = \frac{5}{2^{10}} = 4.883\ mV/bit$$

Teniendo en cuenta que ña caída de tensión máxima en las resistencias shunt comerciales ronda los 75 mV, solo se aprovecharía un 1.5% del rango de medida del ADC.

Ante esto, se debe utilizar otra referencia interna de tensión que el procesador MEGA2560 ofrece, concretamente 1,1V. De esta manera la resolución del ADC mejora significativamente.

$$resolución = \frac{V_{ref}}{2^N} = \frac{1.1}{2^{10}} = 1.074\ mV/bit$$

Esta referencia interna se selecciona mediante código de la siguiente manera:

```
analogReference(INTERNAL1V1);   //Referencia interna del arduino Mega de 1,1V
delay(10);                      //Para estabilizar la placa después del cambio de referencia
```

Una vez definida la referencia de tensión se procede a medir la tensión en las entradas analógicas y calcular la corriente correspondiente.

El cálculo se realiza mediante un bucle for que toma 100 medidas en 1 s para finalmente calcular la corriente media que ha circulado por las baterías durante ese periodo de tiempo. Dentro del bucle for se realizarán las siguientes acciones:

a) Calculo de la diferencia de tensión entre ambos bornes del shunt. Calculando la resta en entre las medidas analógicas de ambos pines de entrada (A0 y A1). Esto devuelve un valor en bits.
b) Se calcula dicha caída de tensión en mV, mediante esta relación:

$$V\ (mV) = \frac{Diferencia\ (bit) * 1{,}1\ V}{1024\ bit} \frac{1000\ mV}{1\ V}$$

c) Se calcula la corriente que corresponde con esa caída de tensión:

$$I\ (A) = V(mV) \frac{I_{max}(A)}{V_{shunt}(mV)}$$

d) Se suma el valor de la corriente medida a la suma del resto de medidas tomadas en el mismo bucle (I_total).
e) Se registra el número de veces que se ejecuta el bucle (N).
f) Se espera 10 ms para espaciar las medidas.

Una vez finalizado el bucle, se calcula la corriente media durante ese periodo de tiempo.

$$I_{dc} = \frac{I_{total}}{N}$$

Es importante resetear tanto el contador como la suma de las corrientes antes de empezar a ejecutar el bucle, para no arrastrar valores de bucles anteriores.

```
//Lectura de la corriente que circula por el shunt
  int valor;
  float corriente;
  float shunt;
  float Idc=0;
  int cont = 0;

  for (int i = 0; i < 100; i ++) //toma 100 medidas por cada segundo y luego se calcula la media
{
  valor= analogRead(A0)-analogRead(A1);   //Diferencia entre los dos bornes del shunt en bits
  shunt = (valor*1.1/1024)*1000;          //1,1V son 1024 bits
  corriente = shunt*100/75;              //100 A son 75mV
  Idc = Idc + corriente;
  cont = cont + 1;
  delay(10);
}
  Idc = Idc/cont;
```

En este punto, se debe definir un criterio de signos claro para definir la dirección de la corriente que circula por el cableado de las baterías. Se decide que dicha corriente sea positiva cuando las baterías se estén cargando.

Al haber conectado el shunt a la salida del polo negativo de la batería, cuando la corriente sea positiva (carga), esta saldrá por el polo negativo de la misma, haciendo que la caída de tensión en el shunt sea positiva si toma de referencia el borne más cercano a la batería.

Por ello, la diferencia de tensión entre ambos pines del ADC se realiza restando a la medida en el pin A0 (cercano a la batería) la medida en el pin A1 (alejado de la batería) y no al revés.

Una vez finalizada la medida de la corriente continua se restablece la referencia interna del convertidor analógico digital para realizare el resto de medidas. Para ello se utiliza el comando analogReference(DEFAULT).

2.5. Medida de la tensión DC

2.5.1. Introducción

Junto con la medida de la corriente que circula por el cableado de las baterías, la medida de la tensión en bornes de estas es necesaria para cuantificar la energía almacenada y cedida por ellas, así como para garantizar un correcto funcionamiento de los equipos, pudiendo detectar cuando la tensión en bornes de las baterías se encuentra fuera del rango aceptable para su utilización.

Todas las baterías utilizadas en fotovoltaica tienen un rango de funcionamiento que va desde la tensión mínima, conocida como *cut off voltaje*, hasta la tensión de carga máxima y es vital para la salud y seguridad de los equipos no exceder ninguno de estos límites.

2.5.2. Elección del método de sensado

El método más empleado para medir voltaje DC es mediante divisores de tensión. Este método se basa en obtener una tensión proporcional a la de la batería pero que se encuentre siempre dentro del rango de medida de la placa.

En esta ocasión no se ha optado por ningún dispositivo comercial y se utiliza un divisor de tensión adaptado a los niveles de tensión de cada una de las baterías que se vayan a medir. Para el dimensionamiento del divisor de tensión se debe tener en cuenta el voltaje de carga máximo, ya que de él depende la ganancia del divisor de tensión.

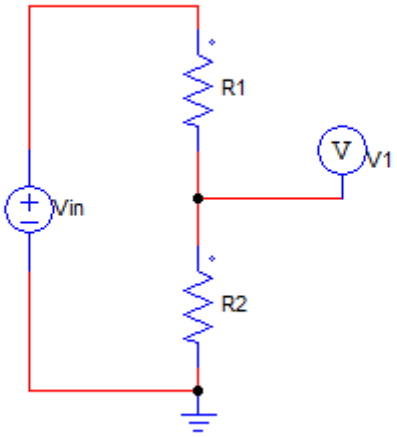


Fig 27 Divisor de tensión

Con esta configuración se obtiene la siguiente relación entre las tensiones:

$$V_{out} = \frac{R_2}{R_1 + R_2} V_{in}$$

De forma que la ganancia del divisor de tensión es:

$$G = \frac{R_2}{R_1 + R_2} = \frac{V_{out}}{V_{in}}$$

Con el fin de aprovechar al máximo el rango de medida del convertidos analógico digital que se utilice se deberá escoger una relación tal entre los valores de las resistencias que haga que la ganancia del divisor de tensión se corresponda entre la tensión máxima de entrada del ADC y la tensión de carga máxima de la batería.

Como se ha indicado anteriormente, el hecho de fabricar el divisor de tensión ofrece la posibilidad de adaptar su ganancia para cada caso. En esta ocasión, se procede a calcular el valor de las resistencias del divisor para una batería de iones de litio de 7 series, siendo la tensión de carga máxima de cada celda 4,2 V. De esta manera, la tensión de carga máxima de la batería será:

$$7 * 4{,}2\ V = 29{,}4\ V$$

Por tanto, sabiendo que la tensión máxima a la entrada del ADC debe ser de 5V, la ganancia del divisor de tensión queda así:

$$G = \frac{R_2}{R_1 + R_2} = \frac{V_{out}}{V_{in}} = \frac{5}{29.4} = 0.17$$

Finalmente, la relación entre ambas resistencias del divisor de tensión deberá ser la siguiente:

$$R_1 = 4.88\ R_2$$

Como los valores comerciales de las resistencias no se adaptan exactamente a esa relación se busca la pareja de resistencias que ofrece la ganancia más cercana 0.17, siendo siempre una ganancia menos, de manera que la tensión máxima medida no supere en ningún caso los 5V. Los valores de las resistencias a utilizar y la ganancia resultante se muestran a continuación:

$$R_1 = 15\ k\Omega$$

$$R_2 = 3\ k\Omega$$

$$G = 0.1667$$

Se comprueba experimentalmente que el divisor de tensión reduce de manera correcta la tensión para adaptarse al rango deseado.

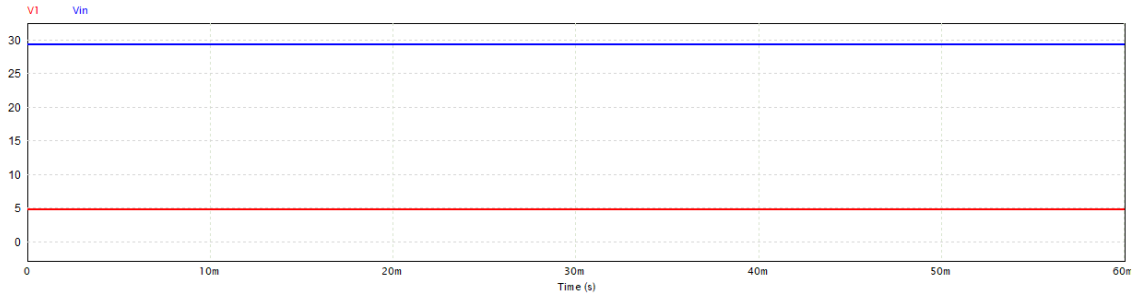


Fig 28 Salida del divisor de tensión

2.5.3. ADS1115

El voltaje de las baterías se mide a través del convertidor analógico-digital externo ADS1115 y no directamente con la placa.

El ADS1115 es un convertidor analógico digital externo de 16 bit, compatible con todo tipo de procesadores. Es un componente creado por la empresa Adafruit que, además de componentes electrónicos, ofrece multitud de librerías de código abierto.

La comunicación entre el ADS1115 y la placa se lleva a cabo mediante el protocolo I2C, a través de los puestos SCL y SDA de la placa.

En comparación con el convertidor ADC interno de la placa ofrece varias ventajas:

g) Mayor resolución. 16 bit del ADS1115 frente a los 10 de la placa.
h) Incorpora un amplificador de ganancia programable, por lo que permite ajustar la resolución del medidor a varias escalas.
i) Es un procesador externo, por lo que reduce el coste computacional de la placa.
j) Aísla el circuito de señal respecto de la placa, protegiéndola ante posibles desequilibrios.
k) Permite lecturas en modo diferencial.

A continuación, se muestra el esquema de conexión del ADS1115 en modo diferencial.

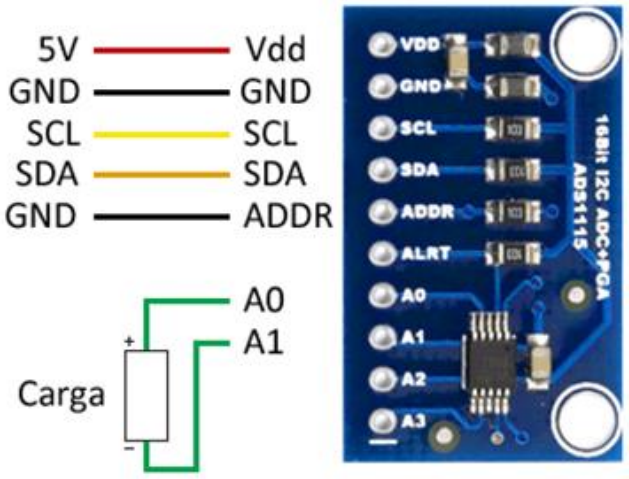


Fig 29 ADS1115

Cabe destacar que esta configuración es válida cuando se utiliza un único ADS1115, en caso de utilizar más de uno ADC externo (hasta 4), deberá modificarse la conexión del pin ADDR (address) e inicializar cada uno de ellos mediante código.

En esta ocasión solo es necesario un único ADS1115, por lo que el pin ADDR se conecta a GND, siendo esta la conexión por defecto.

Cuando se mide en modo diferencial, este ADC calcula la diferencia de tensión entre los pines A1 y A0 (o A3 y A2), tomando como positivo el pin A1 (o A3).

2.5.4. Circuito de medida. Conexión con la placa

El circuito completo para la medida del voltaje de las baterías está compuesto por el divisor de tensión, el convertidor analógico-digital ADS1115 y el bus I2C que envís los datos del ADC externo a la placa.

El voltaje a medir es la caída de tensión producida en bornes de la resistencia R2. Esta tensión se mide de forma diferencial; para ello, se conecta el punto medio del divisor de tensión a la entrada A1 del ADC y el polo negativo de la batería a la entrada A0.

La conexión entre la placa MEGA2560 y el ADS1115 es sencilla, basta con conectar los pines SCL y SDA del ADC con los homónimos de la placa.

2.5.5. Programación

La comunicación entre la placa y el ADS1115 se apoya en las librerías creadas para ello por Adafruit, fabricante del componente. Estas librerías deben ser incluidas al comienzo del script. Además, se debe crear una instancia para cada ADS1115 utilizado.

```
//LIBRERÍAS NECESARIAS PARA EL ADS1115 Y LA COMUNICACIÓN I20
#include <Adafruit_BusIO_Register.h>
#include <Adafruit_I2CDevice.h>
#include <Adafruit_I2CRegister.h>
#include <Adafruit_SPIDevice.h>
#include <Adafruit_ADS1X15.h>
#include <Wire.h>

//Se define la instancia para un ADC
Adafruit_ADS1115 adc;
const float ganancia = 0.1875F; //Depende de la ganancia escogida para cada ADC
//(GAIN_TWOTHIRDS, 6.144V de ref, 0.1875 mV por bit)
```

A continuación, se debe escoger y programar la referencia interna de voltaje del ADC, ya que cuenta con un amplificador de ganancia programable. Para que la reducción necesaria en el divisor de tensión sea la mínima posible, se selecciona siempre el nivel más alto de referencia interna. Esto no supone una pérdida excesiva de resolución, puesto que se trata de un convertidor de 16 bit.

PGA	Referencia (V)	Factor de escala
2/3	6,144 V	0,1875 mV
1	4,096 V	0,1250 mV
2	2,048 V	0,0625 mV
4	1,024 V	0,0312 mV
8	0,512 V	0,0156 mV
16	0,256 V	0,0078 mV

Tabla 3 Ganancias internas del ADS1115

Tal y como indica la anterior imagen, la referencia interna es de 6,144 V y el factor de escala necesario para traducir la lectura (en bits) a voltaje (mV) será 0.1875 mV/bit. Este último valor se ha definido previamente en el apartado de variables como “ganancia”.

En el apartado *void setup* debe inicializarse el ADS1115 y programar la ganancia del amplificador interno.

```
//Configuración del adc externo ADS1115
  adc.begin();                    // Inicializar el adc4 con la dirección por defecto --> pin addr conectado a GND
  adc.setGain(GAIN_TWOTHIRDS);        // +/- 6.144V de referencia --> 1 bit = 0.1875mV
```

Finalmente, se procede al cálculo del voltaje en las baterías. De nuevo, al igual que para la medida de la corriente continua, se realiza mediante un bucle for de 1 s, durante el cual se tomarán 100 medidas. Durante el bucle se realizarán las siguientes acciones:

a) Lectura de la diferencia entre los pines A1 y A0. Esto devuelve un valor en bits que se almacena en la variable “lectura”.
b) Se calcula la diferencia en voltaje (V) entre ambos puntos

$$V\ (V) = \frac{lectura(bit) * ganancia\ \left(\frac{mV}{bit}\right)}{1000}$$

c) Se calcula el voltaje real en bornes de la batería:

$$V_{bat}\ (V) = \frac{V_{adc}(V)}{G}$$

d) Se suma el valor del voltaje de la batería a las medidas obtenidas en el mismo bucle (V_bat).
e) Se registra el número de veces que se ejecuta el bucle (contador).

f) Se espera 10 ms para espaciar las medidas.

Una vez finalizado el bucle, se divide el valor de la suma de todos los voltajes calculados (V_bat) por el número de veces que se ejecutó el bucle (contador), para obtener el valor medio de la tensión en bornes de la batería.

Es importante resetear tanto el contador como la suma de las tensiones antes de empezar a ejecutar el bucle, para no arrastrar valores de bucles anteriores.

```
//VOLTAJE DC
int16_t lectura;
float voltaje;
float V, V_bat;
int contador= 0;
V_bat= 0;

  for (int i = 0; i < 100; i ++) //toma 100 medidas por cada segundo y luego se calcula la media
  {
    lectura = adc.readADC_Differential_0_1();  //lectura del adc en modo diferencial entre A1 y A0
    voltaje = lectura*ganancia/1000;  //Valor de la lectura en voltios (escala 0-5V)
    V = voltaje/G;  //Voltaje real de la batería (5V --> 30V)
    V_bat = V_bat + V;
    contador = contador +1;
    delay(10);
  }

  V_bat = V_bat/contador;
  Serial.print("Vdc= ");
  Serial.print(V_bat);
  Serial.println(" V");
```

En este caso el convenio de signos es evidente, ya que viene se va a medir entre los bornes positivo y negativo de la batería.

CAPÍTULO 3: CÁLCULOS Y PROGRAMACIÓN

3.1. Introducción

En este apartado se describen los cálculos que se deben realizar para caracterizar una instalación fotovoltaica de autoconsumo con o sin baterías. Todos los cálculos se realizarán a partir de las variables calculadas en el capítulo previo. Además, el dispositivo diseñado registrará valores históricos de diversas variables para evaluar la evolución de distintos aspectos de la instalación como el nivel de autoconsumo o de cobertura de la demanda.

Este dispositivo implementará además un sistema de gestión de excedentes con el fin de autoconsumir toda la energía generada posible.

Finalmente, se va a implementar un sistema de avisos visuales (led) como sistema de alarma ante posibles contingencias y como interfaz con el usuario.

3.2. Cálculos

La forma de realizar los cálculos depende de la naturaleza de la instalación, es decir, si se trata de una instalación monofásica o trifásica o de si se cuenta con almacenamiento de energía. A continuación, se describen los cálculos necesarios para cada una de estas combinaciones.

Para la monitorización de la instalación se va a trabajar siempre en términos de energía, por tanto, todas las variables tendrán unidades de Wh o múltiplos de la misma.

En primer lugar, el dispositivo calcula la potencia instantánea en cada uno de los puntos donde se han tomado las medidas. La forma de realizar estos cálculos depende de si la instalación es monofásica o trifásica. A continuación, calcula los flujos de energía a través de las distintas partes de la instalación. Estos últimos cálculos variarán en caso de no existir un sistema de almacenamiento de energía. Para el cálculo de la energía se multiplica la potencia instantánea calculada por el lapso de tiempo que ha transcurrido desde la última medición (en horas); esto es totalmente válido ya que las medidas estarán espaciadas en el tiempo no más de 3 segundos.

Para cuantificar el tiempo que transcurre entre una medida y la siguiente (tiempo que tarda en ejecutarse el *loop* del código) se utiliza un temporizador basado en el comando *millis()* de C++. Para este contador se declaran 3 variables al comienzo del *script*:

```
//SE DEFINEN LAS VARIABLES DEL TEMPORIZADOR
long tiempo = 0;
long tiempototal, tiempoproceso;
```

A continuación, se establece en cero el valor inicial de las variables “tiempototal” y “tiempo”. Para ello, deberá insertarse el siguiente código en el *void setup().*

```
//INICIO DE LOS CONTADORES PARA CALCULO DE ENERGÍA
  tiempototal= 0;                    // Tiempo cuando se empieza a ejecutar cada loop
  tiempo = 0;                        // Tiempo desde que se empezó a ejecutar
```

Al comienzo del *void loop()* se le asigna un valor a las variables “tiempototal” y “tiempo”. La variable “tiempo” toma el valor de *millis(),* es decir, el tiempo total desde que se comenzó a tomar datos y “tiempototal” registra el valor de “tiempo” al inicio de cada *loop*.

```
//DECLARACIÓN DE LOS CONTADORES
 tiempototal = tiempo;
 tiempo = millis();
```

Finalmente, se calcula el tiempo que tarda en ejecutarse el loop como la diferencia entre el valor de "tiempo" y el registrado en "tiempototal" al comienzo del loop. Este cálculo se realiza justo antes del cálculo de los flujos de energía.

```
  tiempoproceso= tiempo-tiempototal;
  float T= tiempoproceso/1000;              //Tiempo del loop en segundos
```

Todos estos valores se calculan en ms, por lo que se declara una variable llamada "T" que registrará dichos tiempos en segundo, por comodidad.

```
float T= tiempoproceso/1000;              //Tiempo del loop en segundos
```

3.2.1. Potencia y energía en las baterías, inversor y acometida.

Tal y como se ha indicado en el primer capítulo de esta memoria, se obtendrán medidas de corriente y voltaje en el cableado de las baterías, si se trata de un autoconsumo con almacenamiento, a la salida del inversor y en la cabecera de la acometida.

Los cálculos de generación, consumo, autoconsumo y demás se realizan a partir del cálculo del flujo de potencia a través de los tres puntos anteriores, traduciendo estos flujos de potencia en energía mediante el temporizador antes diseñado. Es fundamental respetar en todo momento el criterio de signos indicado en el apartado de medición.

a. Flujo de potencia en las baterías

Anteriormente se ha medido tanto la corriente que circula por el cableado del sistema de almacenamiento como el voltaje del mismo. Estos valores se han almacenado en las variables "Idc" y" V_bat" respectivamente.

El sistema de baterías siempre estará integrado en la parte de corriente continua de la instalación, por lo que el cálculo de la potencia instantánea y de la energía que fluye por dicho cableado siempre se realiza de la misma manera, independientemente de si la instalación es monofásica o trifásica.

$$V_{bateria} = I_{dc} * V_{bat}$$

```
//Potencia de las baterias
float Pbat, Ebat;

  Pbat= V_bat * Idc;                      //>0 si es de carga
```

Tal y como se ha definido el criterio de signos para ambas variables, la potencia que circula por el cableado de las baterías será positiva cuando estas se estén cargando y negativo durante las descargas.

b. Flujo de potencia a la salida del inversor

El segundo punto donde el medidor calcula el flujo de potencia es a la salida del inversor. Realmente las medidas de tensión y corriente a la salida del inversor pueden realizarse en

cualquier punto del cableado de salida del inversor hasta el punto de inyección en el cuadro de la instalación, en función de donde se vaya a situar el dispositivo.

Siguiendo el criterio de signos establecido anteriormente, esta potencia será positiva cuando se dirija hacia el consumo. Generalmente esta potencia será siempre positiva, sin embargo, ciertos modelos de inversores ofrecen posibilidad de cargar las baterías desde la red eléctrica.

- Instalaciones monofásicas

En este caso se obtienen únicamente dos variables: "Vinv" e "Iinv".

La potencia que fluye de la instalación fotovoltaica hacia el consumo se calcula de la siguiente manera:

$$P_{inv} = V_{inv} I_{inv} \cos \varphi$$

Para simplificar los cálculos se entiende que el factor de potencia de la instalación es siempre unitario. Esta simplificación se asume como válida ya que los inversores fotovoltaicos habitualmente trabajan con factores de potencia unitarios o muy próximos a la unidad.

```
//Potencia de salida del inversor
float Pinv, Einv;

  Pinv = Vinv*Iinv;                    //Se supone FP=1
```

- Instalaciones trifásicas

En el caso de las instalaciones trifásicas, se obtiene un mayor número de variables: tres tensiones de fase a la salida del inversor ("Vinv1", "Vinv2", "Vinv3") y tres corrientes de fase ("Iinv1", "Iinv2", "Iinv3").

La potencia total se calcula como la suma de la potencia que fluye a través de cada una de las fases.

$$P_{inv} = V_{inv1} I_{inv1} + V_{inv2} I_{inv2} + V_{inv3} I_{inv3}$$

De nuevo, el dispositivo asume que el factor de potencia es unitario en todo momento.

```
//Potencia de salida del inversor
float Pinv, Einv;

  Pinv = Vinv1*Iinv1 + Vinv2*Iinv2 + Vinv3*Iinv3;                    //Se supone FP=1
```

c. Flujo de potencia a la salida de la instalación (acometida)

Finalmente, el dispositivo cuantifica la potencia que se intercambia con la red eléctrica. De nuevo, este cálculo depende de si la instalación es monofásica o trifásica.

Tal y como se ha definido el criterio de signos, el flujo de potencia será positivo cuando energía salga de la instalación hacia la red, es decir, cuando haya excedentes.

- Instalaciones monofásicas

Los valores de voltaje y corriente se han almacenado en las variables "Vred" e "Ired" respectivamente.

La potencia instantánea en la conexión de la instalación con la red eléctrica es la siguiente:

$$P_{red} = V_{red} I_{red} \cos \varphi$$

De nuevo, se asumirá que el flujo de potencia reactiva es nulo y que por tanto el factor de potencia es unitario.

```
//Potencia intercambiada con la red
float Pred, Ered;

  Pred = Vred*Ired;                    //Se supone FP=1
```

- Instalaciones trifásicas

Para instalaciones trifásicas se emplean tres variables de voltaje de fase (“Vred1”, “Vred2”, “Vred3”) y tres de corriente de fase (“Ired1”, “Ired2”, “Ired3”).

La potencia trifásica se calcula como la suma de la potencia monofásica que fluye por cada una de las fases, de nuevo, asumiendo que el factor de potencia es unitario.

$$P_{red} = V_{red1} I_{red1} + V_{red2} I_{red2} + V_{red3} I_{red3}$$

```
//Potencia intercambiada con la red
float Pred, Ered;

  Pred = Vred1*Ired1 + Vred2*Ired2 + Vred3*Ired3;                    //Se supone FP=1
```

3.2.2. Generación fotovoltaica

La energía generada por el generador fotovoltaico puede ser almacenada, en caso de existir un sistema de almacenamiento eléctrico, o transformada e inyectada directamente, en cuyo caso será cuantificada a la salida del inversor.

Por criterio de signos, la energía generada que se amacena en las baterías (carga) será positiva y la energía que sale del inversor hacia el consumo o la red también. De esta manera, la energía fotovoltaica generada se calculará como la suma de ambos conceptos.

Para llevar a cabo dicho cálculo, el dispositivo calcula la suma de la potencia en las baterías y a la salida del inversor y calcula la energía total que se ha generado en el lapso de tiempo que ha trascurrido desde la última medición.

En caso de que las baterías estén entregando potencia a la instalación la variable Pbat tomará un valor negativo, el cual se restará al flujo de potencia que circula a la salida del inversor, obteniendo como resultado únicamente la potencia que fluye directamente desde el generador fotovoltaico.

El dispositivo está programado para calcular la generación eléctrica de la siguiente manera:

```
//GENERACIÓN
//Cuando la potencia en las baterías es positiva (carga) o nula es potencia generada --> Pgen = Pinv + Pbat
//Cuando la potencia en las baterías en negativa (descarga) no es potencia generada --> Pgen = Pinv - Pbat
//Como en ese caso la Pbat es <0 --> P gen = Pinv + Pbat

 Pgen= Pinv + Pbat;

 Egen = Egen + Pgen*T/3600;
```

Finalmente, el valor de la energía generada en el lapso de tiempo transcurrido desde la última medida se almacena en la variable “Egen”.

3.2.3. Excedentes y compra de energía

Cuando la energía generada en la instalación fotovoltaica ni se consume ni se almacena, bien porque no se cuenta con baterías o porque éstas están cargadas al máximo, se exporta a la red eléctrica. Estos excedentes los compensará económicamente la comercializadora del usuario. Por el contrario, cuando la energía generada no es suficiente como para cubrir el consumo instantáneo se importa energía de la red eléctrica.

Siguiendo el criterio de signos establecido anteriormente, cuando la potencia calculada en la acometida es positiva, el flujo de energía es saliente de la instalación, es decir, se está vertiendo energía a la red eléctrica. Por tanto, cuando la potencia en la acometida es negativa, se estará importando energía de la red eléctrica.

El cálculo de Pex (excedentes) y Pimp (importación) se realizará con un bucle if, el cual aplicará la lógica antes comentada. Ambas variables almacenan el valor absoluto.

```
//IMPORTACIÓN, EXCEDENTES y CONSUMO
//Si la potencia de la red es positiva, se está exportando energía (por criterio de signos)
//Si l apotencia de la red es negativa, se está importando energía
  if (Pred >= 0){
    Pex = Pred;
    Pimp = 0;
  }
  else {
    Pimp = -Pred;     //Potencia importada en valor absoluto
    Pex = 0;
  }

  Eex = Eex + Pex*T/3600;
  Eimp = Eimp + Pimp*T/3600;
```

El dispositivo almacena los valores de energía exportada y energía importada desde la última vez que se tomó una medida en las variables “Eex” y “Eimp” respectivamente.

3.2.4. Consumo eléctrico

Una vez obtenido el valor de la energía generada por el generador fotovoltaico y conocido el intercambio de energía con la red eléctrica, el dispositivo calcula fácilmente la energía consumida por el usuario en ese periodo de tiempo.

Cuando la generación sea mayor que la demanda el consumo eléctrico será la diferencia entre la generación y los excedentes y, cuando la demanda sea mayor que la generación, el consumo será la suma de la generación y la energía importada. Como las variables de exportación (Pex) e importación (Pimp) almacenan valores positivos o nulos, el cálculo se realiza siempre de la misma manera, independientemente de si se está exportando o importando energía.

```
//CONSUMO DE ENERGÍA

  Pcons = Pgen - Pex + Pimp;
  Econs = Econs + Pcons*T/3600;
```

El último valor del consumo eléctrico de la instalación queda registrado en la variable “Econs”.

3.2.5. Energía autoconsumida

El cálculo de la energía autoconsumida se realiza de la siguiente manera. Cuando la existen excedentes, la energía autoconsumida coincide con el consumo instantáneo, es decir, será la resta entre la generación y los excedentes. Cuando se está importando energía, la energía autoconsumida coincidirá con la energía generada, sea ésta última nula o no.

```
//AUTOCONSUMO
//Cuando hay excedentes, la energía autoconsumida es la resta entre la generación y los excedentes
//Cuando se importa energía la energía autoconsumida es toda la generada

  if (Pex >0){
    Pautoc = Pgen - Pex;
  }
  else {
    Pautoc = Pgen;
  }
  Eautoc = Eautoc + Pautoc*T/3600;
```

El último valor de energía autoconsumida calculado se registra en la variable "Eautoc".

3.2.6. Porcentaje de autoconsumo y cobertura de la demanda

El objetivo final de este dispositivo es informar de manera intuitiva al usuario de una instalación fotovoltaica sobre los datos instantáneos e históricos de su instalación. una forma de expresar los datos que resulta fácilmente entendible para el usuario es a través de porcentajes.

Por esta razón, el dispositivo registra los datos históricos de generación, consumo, autoconsumo y vertidos para calcular los siguientes porcentajes:

- Porcentaje de autoconsumo

$$\%\, autoconsumo = \frac{Autoconsumo}{Generación} * 100$$

- Porcentaje de excedentes

$$\%\, excedentes = \frac{Vertidos}{Generación} * 100 = 100 - \%\, autoconsumo$$

- Cobertura de la demanda

$$Cobertura\ de\ la\ demanda\ (\%) = \frac{Autoconsumo}{Consumo} * 100$$

- Porcentaje importado

$$\%\, importado = \frac{Consumo - Autoconsumo}{Consumo} * 100 = 100 - Cobertura\ de\ la\ demanda$$

El dispositivo registra los valores históricos y calcula los porcentajes indicados de la siguiente manera:

```
//PORCENTAJE DE AUTOCONSUMO ACUMULADO
//Generación total y consumo total acumulados
  GENERACION= GENERACION + Egen;
  CONSUMO = CONSUMO + Econs;
  AUTOCONSUMO = AUTOCONSUMO + Eautoc;

  porcentaje_autoconsumo = (AUTOCONSUMO/GENERACION)*100;
  porcentaje_excedente = 100 - porcentaje_autoconsumo;

  cobertura_demanda = (AUTOCONSUMO/CONSUMO)*100;
  porcentaje_importado = 100 - cobertura_demanda;
```

3.2.7. Carga de las baterías

Determinar el nivel de carga de las baterías es complejo, ya que no existe ningún indicador mediante el cual se determine directamente la carga de éstas.

Para estimar el nivel de carga de las baterías se contabilizará la energía que almacenan y/o ceden, para conocer en todo momento la carga almacenada en estas. Esta técnica necesita conocer el valor inicial de la carga, lo cual habitualmente no es posible. En ese caso, se estimará que la carga inicial de las baterías es el 100% de su capacidad y se limitará el valor de la carga almacenada al dicho valor máximo. De esta manera, cuando la carga real alcance el 100% de la capacidad de las baterías por primera vez, el valor de la carga calculada por el dispositivo será fiel a la realidad.

```
//ESTADO DE CARGA DE LAS BATERÍAS
//Se calcula el nivel de carga de las baterías en función de la energía almacenada

  Ebat = Ebat + Pbat*T/3600;

  if (Ebat > cargamax) {
    Ebat = cargamax;}

  porcentaje_bateria = (Ebat/cargamax)*100;
```

3.3. Gestión de excedentes

Los excedentes son compensados económicamente por la comercializadora, lo cual reduce el importe del término variable en la factura eléctrica del usuario. Sin embargo, estos excedentes se compensan según el precio del mercado horario de la energía (*pool*) el cual es notablemente menor que el precio que los usuarios pagar por la energía. Es por ello que resulta interesante autoconsumir al máximo la energía producida.

Además, la rentabilidad y el periodo de amortización de la inversión dependerán directamente del grado de autoconsumo.

Con el fin de autoconsumir la mayor energía producida posible, el dispositivo implementa tres salidas monofásicas que se activarán siempre que la producción de energía sea mayor que la demanda. Estas salidas están pensadas para alimentar cargas que no necesiten una alimentación continua (sistemas de extracción de aire, aires acondicionados) o que transformen la electricidad en otro tipo de vector energético como agua caliente (termos eléctricos). Estas salidas permiten hibridar las instalaciones de autoconsumo eléctrico con instalaciones de calefacción, agua caliente sanitaria, aerotérmicas u otras formas de aprovechamiento de la energía eléctrica generada.

Los relés que incorpora el dispositivo son de estado sólido. Este tipo de relés no tiene partes móviles que se desgasten en las conmutaciones, por lo que no requieren ningún tipo de mantenimiento y su vida útil es larga.

La programación para activar estas salidas es sencilla, cuando existen excedentes se activarán los relés que habilitan las tres salidas. Además, junto con los relés se activa un led conectado al pin digital 13 para indicar que las salidas están habilitadas.

```
//GESTIÓN DE EXCEDENTES
  if (Pex >0){
    //Cuando hay excedentes se activa la salida del relé
    digitalWrite(rele_r, HIGH);
    digitalWrite(rele_s, HIGH);
    digitalWrite(rele_t, HIGH);
    digitalWrite(13, HIGH);
    }
  else {
    digitalWrite(rele_r, LOW);
    digitalWrite(rele_s, LOW);
    digitalWrite(rele_t, LOW);
  }
```

3.4. Avisos y alarmas

El dispositivo implementa un sistema de avisos visuales mediante bombillas led que sirve como interfaz para el usuario y como sistema de alarma en caso de desequilibrio en la tensión de una de las fases.

El sistema de avisos se ha implementado mediante leds y no con una pantalla lcd porque se trata de un complemento a la información transmitida a través de la página web, por lo que no merece la pena asumir el consumo de energía que la pantalla lcd supondría.

3.4.1. Carga de las baterías

Para indicar el estado de carga de la batería se utilizan 5 bombillas led azules. El número de bombillas encendidas simultáneamente dependerá de la carga de la batería, para ello el dispositivo sigue la siguiente secuencia:

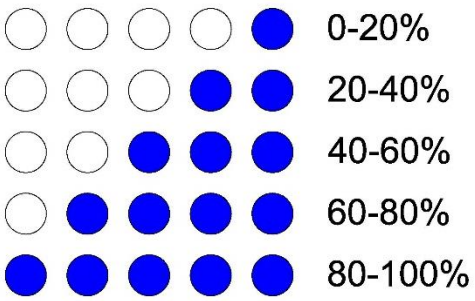


Fig 30 Código de colores para indicador de carga de las baterías

Cuando la batería esté totalmente descargada se apagarán todos los leds.

Para esta función, el dispositivo está programado de esta manera:

```
if (porcentaje_bateria > 80){
  digitalWrite(carga20, HIGH);
  digitalWrite(carga40, HIGH);
  digitalWrite(carga60, HIGH);
  digitalWrite(carga80, HIGH);
  digitalWrite(carga100, HIGH);
  }
else if (80 >= porcentaje_bateria > 60){
  digitalWrite(carga20, HIGH);
  digitalWrite(carga40, HIGH);
  digitalWrite(carga60, HIGH);
  digitalWrite(carga80, HIGH);
  digitalWrite(carga100, LOW);
} else if (60 >= porcentaje_bateria > 40){
  digitalWrite(carga20, HIGH);
  digitalWrite(carga40, HIGH);
  digitalWrite(carga60, HIGH);
  digitalWrite(carga80, LOW);
  digitalWrite(carga100, LOW);
}


  else if (40 >= porcentaje_bateria > 20){
  digitalWrite(carga20, HIGH);
  digitalWrite(carga40, HIGH);
  digitalWrite(carga60, LOW);
  digitalWrite(carga80, LOW);
  digitalWrite(carga100, LOW);
}
  else if (20 >= porcentaje_bateria ){
  digitalWrite(carga20, HIGH);
  digitalWrite(carga40, LOW);
  digitalWrite(carga60, LOW);
  digitalWrite(carga80, LOW);
  digitalWrite(carga100, LOW);
}
else if (porcentaje_bateria = 0){
  digitalWrite(carga20, LOW);
  digitalWrite(carga40, LOW);
  digitalWrite(carga60, LOW);
  digitalWrite(carga80, LOW);
  digitalWrite(carga100, LOW);
}
```

3.4.2. Alarma por desequilibrio de tensión entre las fases del sistema

El hecho de medir todas las tensiones de fase, tanto a la salida del inversor como en la acometida de la instalación, permite comprobar si la carga conectada a cada una de las fases está equilibrada o si el inversor ofrece una salida equilibrada entre las fases.

El desequilibrio entre las fases se debe generalmente a las cargas monofásicas y a un reparto inadecuado de estas entre las distintas fases del sistema, además, un desequilibrio de tensión a la salida del inversor fotovoltaico puede evidenciar un mal funcionamiento del mismo.

El desequilibrio de tensión entre las fases del sistema se calcula de la siguiente manera:

$$\mu\,(\%) = \frac{\max(|V_r - V_s|, |V_r - V_t|, |V_s - V_t|)}{\frac{V_r + V_s + V_t}{3}}$$

Este desequilibrio no debe superar en ningún momento el 2% para garantizar la seguridad y el correcto funcionamiento de los equipos conectados a dicha red.

El dispositivo calcula el desequilibrio y enciende los leds siguiendo el siguiente código:

```
//DESEQUILIBRIO DE FASES
//Desequilibrio a la salida del inversor
float Vmedia = (Vinv1 + Vinv2 + Vinv3)/3;
float desequilibrio;

  desequilibrio = (max(max(abs(Vinv1-Vinv2),abs(Vinv1-Vinv3)),abs(Vinv2-Vinv3))/Vmedia)*100;

//Desequilibrio a la salida del inversor
float Vmedia_red = (Vred1 + Vred2 + Vred3)/3;
float desequilibrio;

  desequilibrio_red = (max(max(abs(Vred1-Vred2),abs(Vred1-Vred3)),abs(Vred2-Vred3))/Vmedia_red)*100;

  if (desequilibrio > 2){
    digitalWrite(l1, HIGH); //Enciende el led de alarma
  }else{
    digitalWrite(l1, LOW);
  }
  if (desequilibrio > 2){
    digitalWrite(l2, HIGH); //Enciende el led de alarma
  }else{
    digitalWrite(l2, LOW);
  }
```

CAPÍTULO 4: MONTAJE Y COLOCACIÓN DEL DIPOSITIVO

El montaje del dispositivo se ha realizado de acuerdo al esquema eléctrico del ANEXO I.

4.1. Montaje experimental

El montaje experimental, durante la fase de pruebas se realizó mediante placas de conexión, conocidas como *protoboard.* Estas placas permiten montar y desmontar fácilmente circuitos electrónicos y son muy útiles durante las fases de diseño.

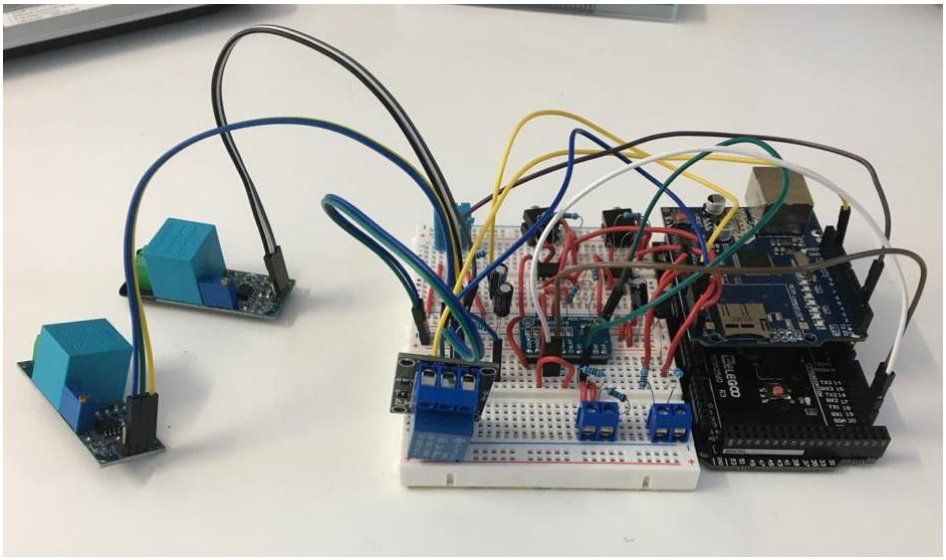


Fig 31 Montaje experimental

Sin embargo, estas conexiones no son fiables para un producto final, por lo que se ha optado por otra técnica de conexión parta el dispositivo final.

4.2. Montaje definitivo

El montaje definitivo del dispositivo se ha realizado mediante placas de soldadura y soldadura manual con estaño. Todos los sensores y circuitos de adecuación de señal se han implementado dentro de una caja, en cuyos laterales se realiza la conexión de los cableados y sensores externos para cada una de las medidas.

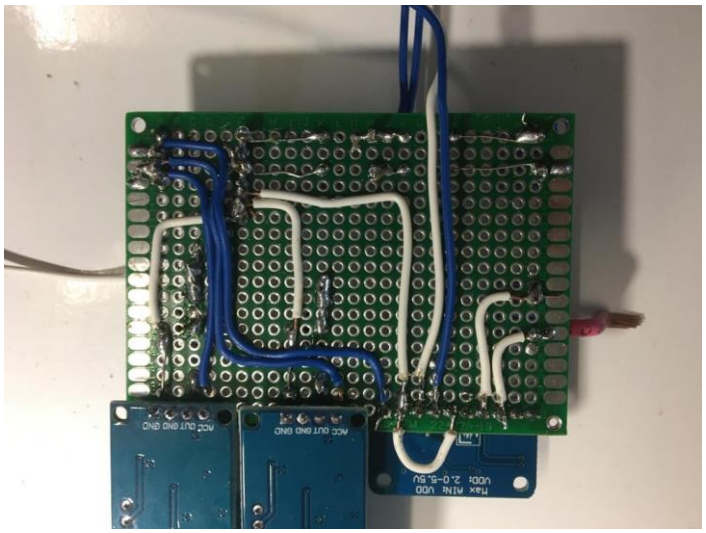


Fig 32 Detalle de la técnica de soldadura

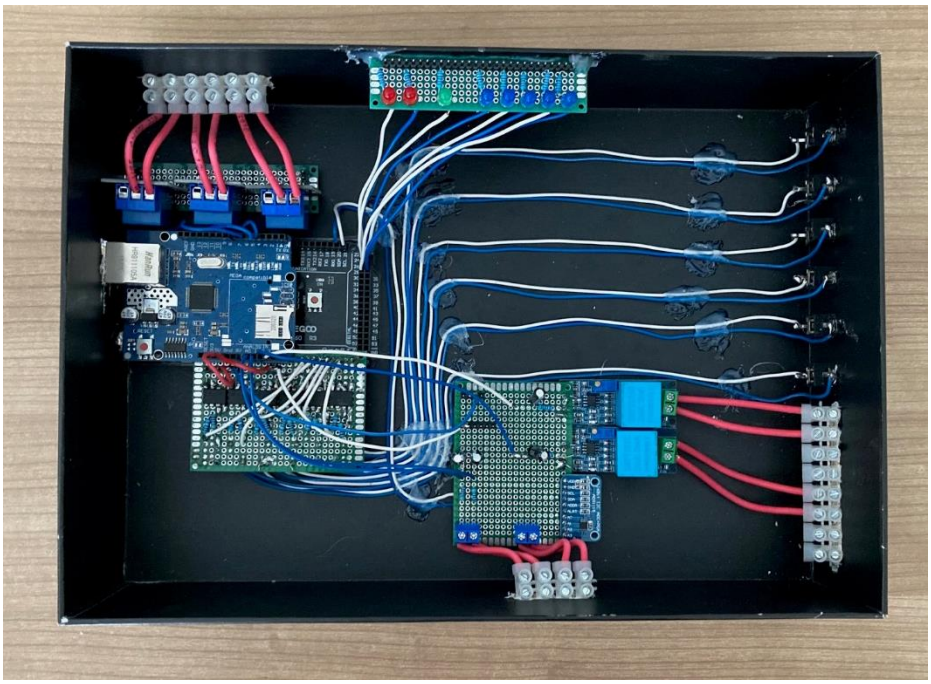


Fig 33 Montaje definitivo sin tapa

Fig 34 Montaje definitivo con tapa

Las conexiones de los sensores y salidas se realizan mediante regletas, excepto en el caso de las pinzas amperimétricas que se conectan a través de conectores Jack hembra.

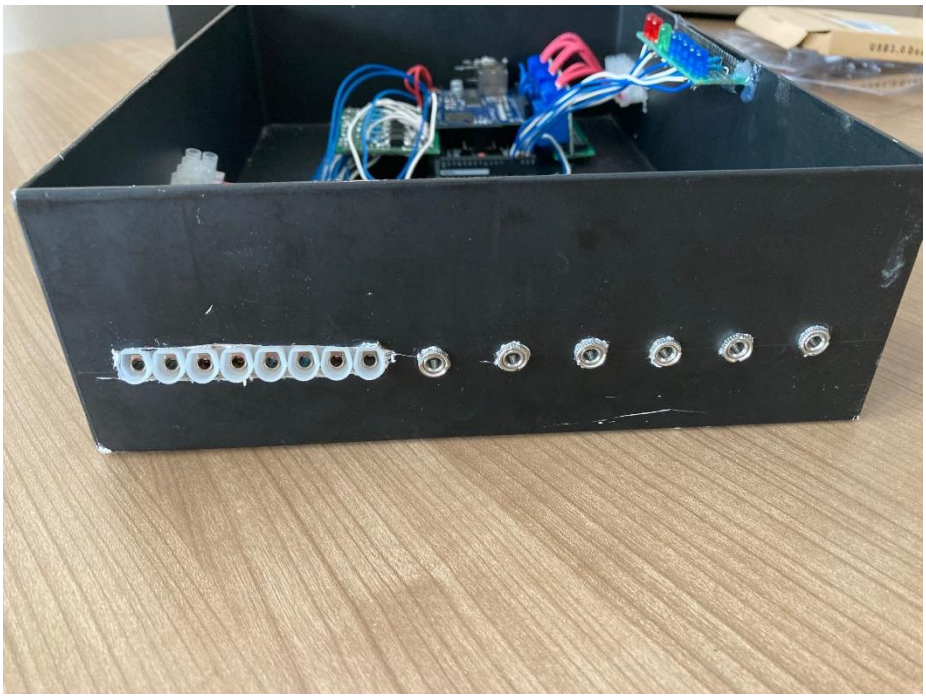


Fig 35 Detalle de las conexiones de los sensores de voltaje y corriente AC

4.3. Colocación

Este dispositivo está diseñado para colocarse cerca del cuadro de protecciones de la instalación, donde se realiza tanto la conexión con la acometida como la inyección en red interior de la generación fotovoltaica. De esta manera, se colocarán fácilmente las pinzas amperimétricas y los cables que conectarán cada una de las fases con la regleta del dispositivo.

Habitualmente, las baterías se colocarán cerca del cuadro de la instalación, por lo que resultará fácil conectar el cableado de medida de las baterías con el dispositivo.

- Colocación de la resistencia shunt y medida del voltaje de las baterías

Para la medida de la tensión en las baterías se debe conectar cada borne (positivo y negativo) con los bornes de la regleta del dispositivo preparados para ello. Ver figura 36.

Para respetar el criterio de signos establecido para la programación del cálculo de los resultados, el sensor de corriente continua se debe colocar siempre en el cableado que sale del borne negativo de las baterías.

Esta conexión es independiente de la topología de la instalación, es decir, es indiferente que la instalación cuente con un regulador de carga y un inversor o que se utilice un inversor/regulador/cargador (3 en 1), ya que el cableado de las baterías siempre será el mismo.

Una vez colocado el sensor en el cableado de la instalación se conectarán ambos extremos de la resistencia con la regleta de entrada al dispositivo de la siguiente manera

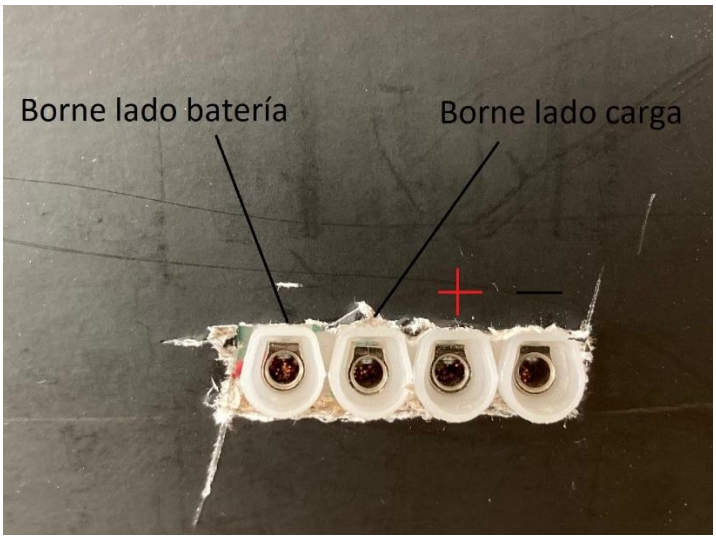


Fig 36 Conexión del shunt y medida de voltaje de las baterías

- Colocación las pinzas amperimétricas a la salida del inversor

Es fundamental colocar las pinzas en el sentido correcto para respetar el criterio de signos. En el caso de la medida de la corriente generada, se debe colocar la flecha del sensor en dirección al consumo, siguiendo la dirección del flujo de energía generada. En las instalaciones trifásicas se colocarán todas las pinzas en el mismo sentido.

Fig 37 Foto detalle del sensor SCT-013-030

La conexión de las pinzas con el dispositivo se realizará de acuerdo a la figura 38.

- Colocación de las pinzas amperimétricas en la acometida

De nuevo, se deberá respetar la dirección de las pinzas para cumplir con el criterio de signos definido. En esta ocasión la flecha de los sensores deberá dirigirse hacia el exterior de la instalación, es decir, hacia la red eléctrica.

La conexión de las pinzas con el dispositivo se realizará de acuerdo a la figura 38.

- Medida de las tensiones de fase

Para la medida de las tensiones de fase se conectará un cable desde cada punto que se debe medir hasta los terminales del dispositivo preparados para ello. En la siguiente imagen se indica donde se debe realizar cada una de las conexiones.

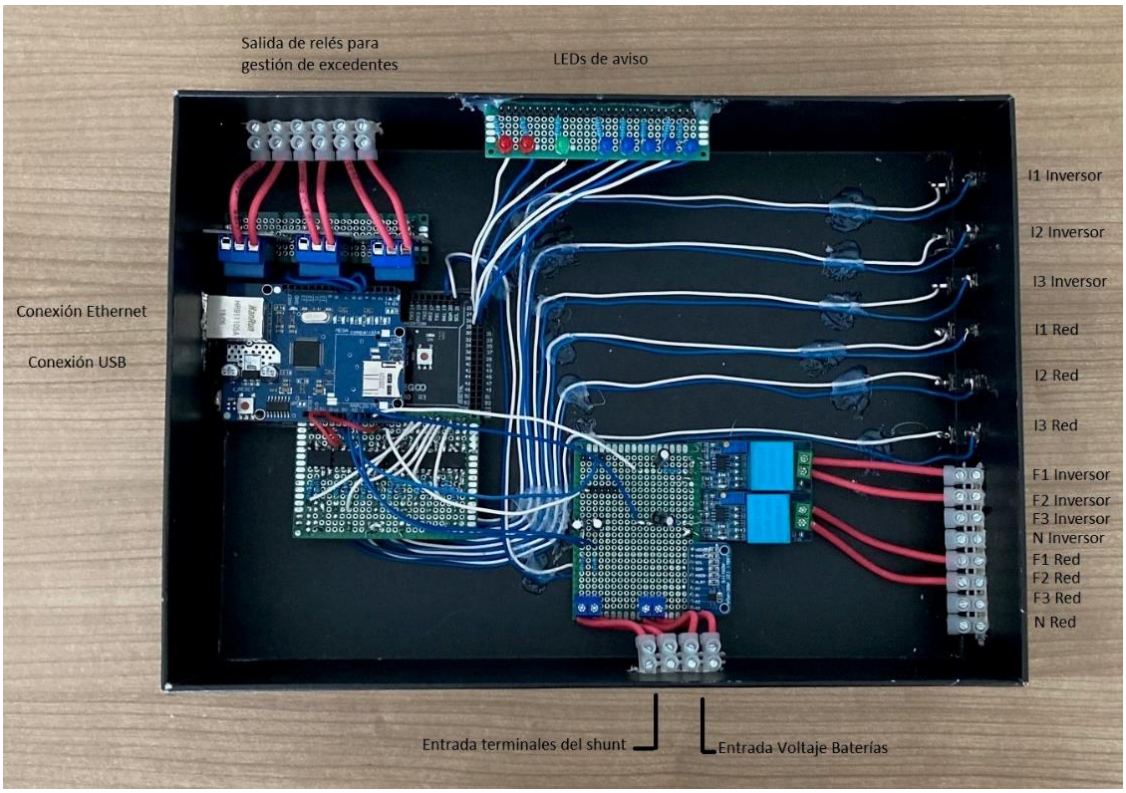


Fig 38 Conexiones del dispositivo

CAPÍTULO 5: PUBLICACIÓN DE LOS DATOS OBTENIDOS

5.1. Introducción

Los datos obtenidos y calculados hasta el momento se envían desde la placa a un servidor web donde se almacenan y publican a través de una página web mediante la cual el usuario tiene acceso a ellos tanto en formato numérico como de forma gráfica.

En este apartado se describe el funcionamiento del sistema de publicación y almacenamiento de los datos, el proceso de diseño de la página web y la interfaz final mediante la cual el usuario accede a sus datos.

5.2. Dominio

El primer paso para empezar a crear una página web propia es contratar y registrar el dominio con el cual se identificará la página. Los dominios se contratan en internet y existe gran cantidad de proveedores de dominios y alojamientos web (hosting). Habitualmente se ofrece la posibilidad de contratar ambos servicios a la vez, es decir, un dominio para la web y espacio en un servidor web (hosting) para alojar la misma.

A la hora de elegir el domino se debe escoger tanto el nombre del mismo como el tipo de dominio (".com", ".es", ".net", p.ej.). Los dominios ".com" ofrecen generalmente mejor posicionamiento web que los dominios nacionales o regionales del tipo ".es". El dominio escogido y contratado para publicar los datos del dispositivo es el siguiente:

www.mienergiafotovoltaica.com

En este caso, no se precisa contratar alojamiento en un servidor web, por lo que se contrató únicamente un dominio. Generalmente, contratar un dominio supone un gasto muy reducido, sin embargo, contratar un alojamiento web puede suponer un coste considerable, dependiendo de los servicios que este ofrezca.

5.3. Alojamiento en servidor web

Las páginas web deben estar alojadas en un servidor, el cual almacena toda la información necesaria para su funcionamiento. Entre la información que almacena el servidor destaca la base de datos asociada a la página, la información de la interfaz de usuario (index), los archivos de recepción de datos externos (necesarios para recibir datos desde Arduino) y los plugins o herramientas instaladas para cualquiera de las funcionalidades la misma.

El almacenamiento se puede realizar en un servidor propio o contratar un servicio de alojamiento externo en la nube. Si se trabaja con un servidor local se ahorran los gastos que supone el servicio externo, sin embargo, el proceso de configuración y mantenimiento del servidor es complejo y requiere un gran conocimiento de la materia. En caso de contratar un servicio externo, se obtendrá una capacidad de almacenamiento limitada (generalmente suficiente) y un número de webs limitado, o no, dependiendo del contrato, eso sí, estos servicios ofrecen interfaces de usuario intuitivas y un servicio de atención al cliente que facilita significativamente el trabajo.

Para este proyecto se cuenta con el servicio de alojamiento web de la empresa Energías Fotovoltaicas de Navarra, en el cual se encuentran alojadas las páginas web del grupo FOTONA. Se trata de un espacio de almacenamiento gestionado por la empresa Your Concept Group. Para

este proyecto se dispone de alojamiento para el dominio contratado y de 16 GB de almacenamiento.

Para enlazar el dominio con el servidor, se debe introducir las direcciones DNS del servidor web en el panel de gestión del proveedor del dominio.

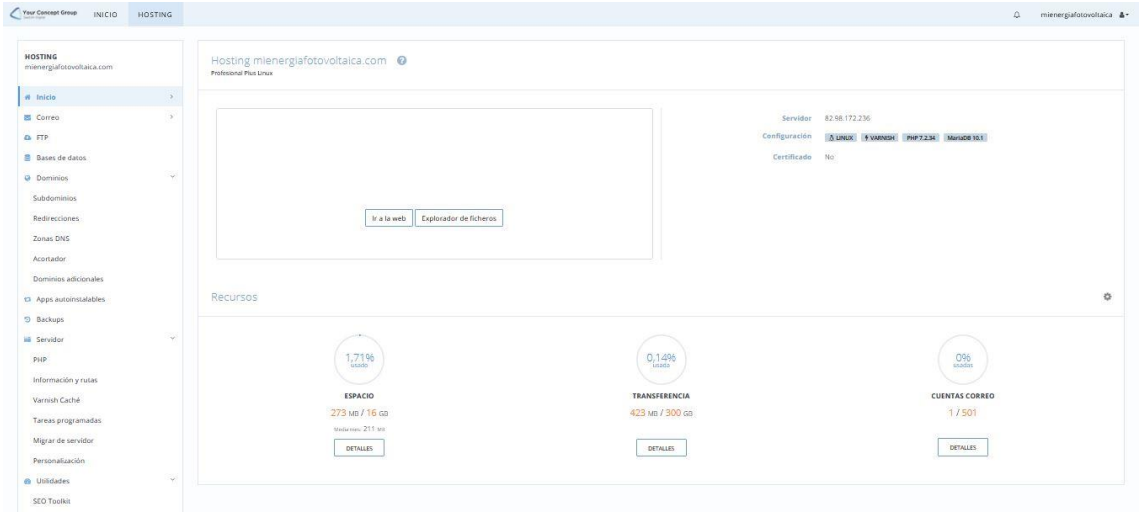


Fig 39 Panel de administración del servidor web

5.4. Base de datos

Antes de enviar y recibir los datos desde el dispositivo al servidor web, se configuró la base de datos donde se almacenan los datos. Como se ha indicado anteriormente, esta base de datos está alojada en el servidor web, lo cual equivale a un servicio de almacenamiento de datos en la nube.

Este servicio de hosting utiliza MariaDB 10.1 como sistema de gestión de bases de datos. Este sistema es completamente compatible con MySQL y permite gestionar todo tipo de bases de datos SQL. Además, el servicio de hosting incluye la herramienta phpMyAdmin, la cual permite diseñar y gestionar bases de datos escritas en php de forma intuitiva sin necesidad de programarlas mediante código.

5.4.1. Diseño de la base de datos

El primer paso para diseñar la base de datos es decidir los datos que se vana a almacenar. El dispositivo calcula y registra numerosas variables, sin embargo, la mayoría de ellas son variables intermedias utilizadas para realizar los cálculos. Las variables necesarias para representar los datos en la web y por tanto las que se almacenan en la base de datos son:

- Generación.
- Consumo.
- Autoconsumo.
- Vertido.
- Importado.
- Porcentaje de autoconsumo.
- Porcentaje excedente.
- Cobertura de la demanda.
- Porcentaje importado.

- Porcentaje de carga de las baterías

Además, cuando se reciban los datos, a cada fila de la tabla se le asignará un "id" (identificador), que será un número incremental para evitar duplicados. También se registrará la fecha y la hora a la que se reciben los datos.

Finalmente, la estructura de la tabla donde se almacenen los datos será la siguiente:

Id	Fecha	Hora	Gen	Cons	Autoc	Vert	Imp	%Aut	%Ex	Cob	%Imp	Carga

Tabla 4 Tabla para almacenar datos en base de datos

5.4.2. Creación de la base de datos

En el apartado "Bases de datos" del panel de gestión de hosting de Your Concept Group, se crea la base de datos a la que se le asigna un usuario y contraseña para el administrador de la misma.

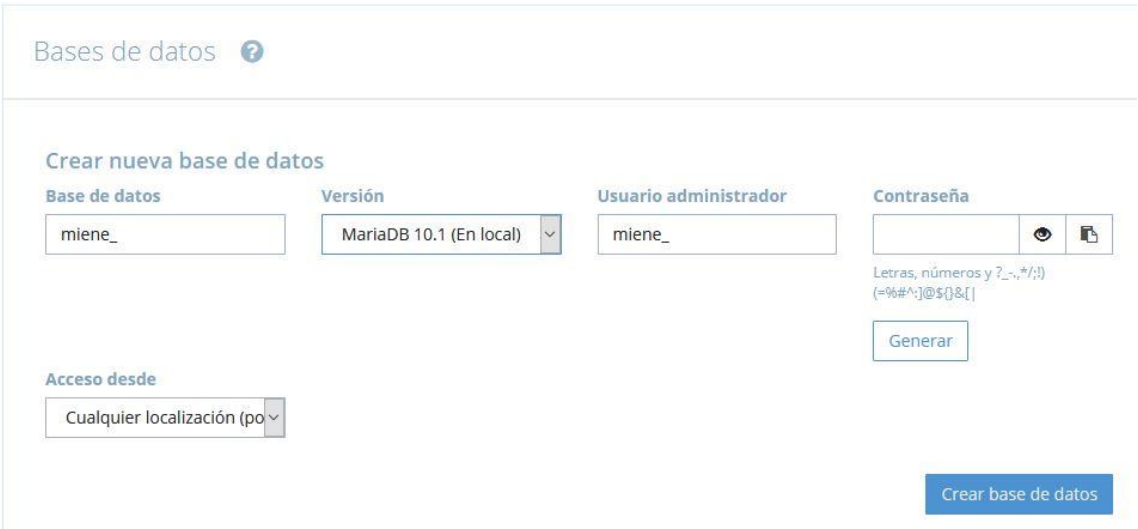


Fig 40 Creación de bases de datos

Para este proyecto se ha creado una base de datos llamada "registro_xabi".

Una vez creada la base de datos, se debe crear una tabla con las columnas señaladas en el apartado anterior. Esta configuración se realiza forma manual con la herramienta phpMyAdmin. A la hora de crear la tabla se debe escoger el tipo de datos que se almacenaran en cada una de las columnas. La tabla creada para almacenar los datos se llama "resultados_finales" y tiene las estas características.

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
1	id	int(11)			No	*Ninguna*		AUTO_INCREMENT	
2	fecha	date			Sí	*NULL*			
3	hora	time			Sí	*NULL*			
4	generacion	float(6,2)			Sí	*NULL*			
5	consumo	float(6,2)			Sí	*NULL*			
6	autoconsumo	float(6,2)			Sí	*NULL*			
7	vertido	float(6,2)			Sí	*NULL*			
8	importado	float(6,2)			Sí	*NULL*			
9	porcentaje_autoconsumo	float(6,2)			Sí	*NULL*			
10	porcentaje_excedente	float(6,2)			Sí	*NULL*			
11	cobertura_demanda	float(6,2)			Sí	*NULL*			
12	porcentaje_importado	float(6,2)			Sí	*NULL*			
13	porcentaje_bateria	float(6,2)			Sí	*NULL*			

Fig 41 Propiedades de la tabla

5.5. Envío de datos

El dispositivo realiza el envío de datos a través de ethernet, pasando por el router de la instalación. Este método de envío es mucho más fiable que el envío mediante wifi, ya que asegura la continuidad de la conexión en todo momento.

Para el envío de datos se utiliza el módulo Ethernet Shield W5100. Este es un componente diseñado para ser utilizado con placas de expansión que implementen cualquiera de los microprocesadores de la gama Atmel, entre ellos el AtMega2560. El Ethernet Shield W5100 es un dispositivo que realiza la conexión entre el microprocesador y la red internet a través del protocolo de conexión TCP/IP.

La toma de datos se realiza con una periodicidad de unos pocos segundos, por lo que es impensable realizar el envío de los datos cada vez que se ejecuta el *loop*. El dispositivo almacena los datos obtenidos durante 5 minutos y realiza el envío de dichos datos cada vez que transcurre ese tiempo. De esta manera, se realizan 12 envíos de datos cada hora, 288 registros al día. En ningún momento se pierde información y se reduce el coste computacional tanto de la placa como del servidor que recibe los datos.

El envío de datos a un servidor web se puede realizar de dos maneras: con consultas html tipo GET o tipo POST. En este caso, se ha optado por consultas html de tipo GET. Este método de envío resulta más ligero para el procesador que las consultad de tipo POST, por lo que se reduce el coste computacional. Las consultas tipo GET pueden resultar problemáticas cuando se realiza la comunicación con servidores ajenos, por cuestiones de permisos. Como el servidor al que se envían los datos es propio, las consultas GET no presentan ningún impedimento.

Los datos enviados deben llegar al archivo “databaseconnect.php” almacenado en la raíz del servidor, ya que este se encargará de almacenar los datos en la base de datos. La ruta hasta dicho archivo es la siguiente:

http://www.mienergiafotovoltaica.com/databaseconnect.php

El dispositivo está programado de la siguiente manera para realizar el envío de datos:

- Declaración de las librerías y creación de la instancia para el Ethernet Shield

```
//LIBRERÍAS NECESARIAS PARA EL ENVIO DE DATOS CON ETHERNET AL SERVIDOR
#include <SPI.h>
#include <Ethernet.h>

EthernetClient client;

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
char serverName[] = "www.mienergiafotovoltaica.com";    //Nombre del dominio propio
int serverPort = 80;    //Puerto para la transmisión de datos
char pageAdd[64];
int totalCount = 0;
int FrecuenciaEnvio= 300000; //5 min = 300000 ms
```

- Inicialización del shield

```
Serial.print(F("Starting ethernet..."));
if(!Ethernet.begin(mac)) Serial.println(F("failed"));
else Serial.println(Ethernet.localIP());

delay(2000);
Serial.println(F("Ready"));
```

- Cada 5 minutos se llama al conjunto de funciones que realiza el envío de datos y se resetean las variables que acumulan los datos.

Para el envío de datos se llama a la orden getPage() que se ejecuta al final del script.

```
if (tiempototal > FrecuenciaEnvio){
 //ENVÍO DE LOS DATOS ALMACENADOS EN LOS ÚLTIMOS 5 MIN
 //Se envía Egen, Econs, Eimp, Eex, Eautoc, porcentaje_autoconsumo,
 //porcentaje_excedente, cobertura_demanda, porcentaje_importado y porcentaje_bateria

   sprintf(pageAdd,"/databaseconnect.php?lectura=%s",totalCount);

 //SI FALLA LA CONEXIÓN AVISA, SI NO SE LLAMA getPage
   if(!getPage(serverName,serverPort,pageAdd)) Serial.print(F("Fail "));
   else Serial.print(F("Pass "));
   totalCount++;
   Serial.println(totalCount,DEC);

 //RESETEO DE LOS VALORES DE ENERGÍA ACUMULADOS
 Egen =0;
 Eimp =0;
 Eex =0;
 Econs =0;
 Eautoc =0;

}
```

- Envío de los datos. getPage()

```
//ENVÍO DE LOS DATOS
byte getPage (char* domainBuffer,int thisPort,char* page)
{ int inChar;
  char outBuf[64];
  Serial.print(F("connecting..."));
  if(client.connect(domainBuffer,thisPort) == 1)
  { Serial.print(F("connected to "));
    Serial.println(domainBuffer);
    Serial.println();
    client.print("GET /databaseconnect.php?generacion=");
    client.print(Egen);
    client.print("&consumo=");
    client.print(Econs);
    client.print("&autoconsumo=");
    client.print(Eautoc);
    client.print("&exportado=");
    client.print(Eex);
    client.print("&importado=");
    client.print(Eimp);
    client.print("&porcentaje_autoconsumo=");
    client.print(porcentaje_autoconsumo);
    client.print("&porcentaje_excedente=");
    client.print(porcentaje_excedente);
    client.print("&cobertura_demanda=");
    client.print(cobertura_demanda);
    client.print("&porcentaje_importado=");
    client.print(porcentaje_importado);
    client.print("&porcentaje_bateria=");
    client.print(porcentaje_bateria);
    client.println(" HTTP/1.1");

    sprintf(outBuf,"Host: %s",serverName);
    client.println(outBuf);
    client.println(F("Connection: close\r\n")); }
  else
  { Serial.println(F("failed"));
    return 0; }
  int connectLoop = 0;
  while(client.connected())
  {
    while(client.available())
    {
      inChar = client.read();
      Serial.write(inChar);
      connectLoop = 0; }
    delay(1);
    connectLoop++;
    if(connectLoop > 10000)
    {
      Serial.println();
      Serial.println(F("Timeout"));
      client.stop(); } }
  Serial.println();
  Serial.println(F("disconnecting."));
  client.stop();
  return 1;
}
```

Para que la conexión entre la placa y el servidor se ejecute se debe inhabilitar el protocolo de seguridad https del servidor, ya que el microprocesador AtMega2560 no soporta el coste computacional que este protocolo requiere.

5.6. Recepción y almacenamiento de datos

La recepción de datos se realiza mediante un código php que recoge la consulta html de tipo GET y desglosa todas las variables enviadas. Como se ha indicado anteriormente, este archivo se llama “databaseconnect.php” y está almacenado en la raíz del servidor.

Este archivo actúa como intermediario entre el dispositivo y la base de datos. En primer lugar, recoge e interpreta la consulta html realizada por el dispositivo y en segundo lugar, almacena los datos recibidos en la base de datos mediante una consulta PDO.

- Conexión con la base de datos

En primer lugar, se abre la conexión con la base de datos. Para ello, se ejecuta el archivo “pdoconfing.php”, el cual contiene la información y los comandos necesarios para establecer la conexión con la base de datos. El archivo “pdoconfig.php” contiene lo siguiente:

```php
<?php
    $host = 'localhost';
    $dbname = 'registro_xabi';
    $username = 'xabi';
    $password = '**************';
?>
```

- Recepción y registro de los datos en la base de datos

Los datos recibidos se almacenan en la base de datos mediante el siguiente código PHP. Para ello, se utiliza la extensión de objetos de datos PDO (PHP Data Objects), una herramienta de PHP para la gestión ligera de bases de datos.

```php
<?php
//SE LLAMA AL ARCHIVO “pdoconfig.php” PARA LA CONEXIÓN CON LA BD
require_once 'pdoconfig.php';
//SE ESTABLECE LA CONEXIÓN CON LA BD
 try {
   $conn = new PDO("mysql:host=$host;dbname=$dbname", $username, $password);
   echo "Connected to $dbname at $host successfully.";
} catch (PDOException $pe) {
   die("Could not connect to the database $dbname :" . $pe->getMessage());
```

```
}
//SE DECLARA LA ZONA HORARIA Y SE RECOGEN LAS VARIABLES FECHA Y HORA
ini_set('date.timezone', 'Europe/Madrid');
$fecha = date("Y/m/d");
$hora = date("H:i:s");
//SE RECIBEN LAS VARIABLES ENVIADAS POR GET HTML
$generacion = $_GET['generacion'];
$consumo = $_GET['consumo'];
$autoconsumo= $_GET['autoconsumo'];
$importado = $_GET['importado'];
$exportado = $_GET['exportado'];
$porcentaje_autoconsumo = $_GET['porcentaje_autoconsumo'];
$porcentaje_excedente= $_GET['porcentaje_excedente'];
$cobertura_demanda = $_GET['cobertura_demanda'];
$porcentaje_importado = $_GET['porcentaje_importado'];
$porcentaje_bateria = $_GET['porcentaje_bateria'];

//SE INSERTAN LOS DATOS EN LA BASE DE DATOS CON PDO

$sql="insert into resultados_finales(fecha, hora, generacion, consumo, autoconsumo, vertido, importado, porcentaje_autoconsumo, porcentaje_excedente, cobertura_demanda, porcentaje_importado, porcentaje_bateria)
values(:fecha, :hora, :generacion, :consumo, :autoconsumo, :exportado, :importado, :porcentaje_autoconsumo, :porcentaje_excedente, :cobertura_demanda, :porcentaje_importado, :porcentaje_bateria)";
//SE PREPARA UNA CONSULTA PDO
$sql = $conn->prepare($sql);

//SE ENLAZEN LOS MARCADORES DE POSICIÓN A LAS VARIABLES
$sql->bindParam(':fecha',$fecha,PDO::PARAM_STR, 25);
$sql->bindParam(':hora',$hora,PDO::PARAM_STR, 25);
$sql->bindParam(':generacion',$generacion,PDO::PARAM_STR, 25);
```

```
$sql->bindParam(':consumo',$consumo,PDO::PARAM_STR, 25);

$sql->bindParam(':autoconsumo',$autoconsumo,PDO::PARAM_STR, 25);

$sql->bindParam(':exportado',$exportado,PDO::PARAM_STR, 25);

$sql->bindParam(':importado',$importado,PDO::PARAM_STR, 25);

$sql>bindParam(':porcentaje_autoconsumo',$porcentaje_autoconsumo,PDO::PARAM_STR, 25);

$sql->bindParam(':porcentaje_excedente',$porcentaje_excedente,PDO::PARAM_STR, 25);

$sql->bindParam(':cobertura_demanda',$cobertura_demanda,PDO::PARAM_STR, 25);

$sql->bindParam(':porcentaje_importado',$porcentaje_importado,PDO::PARAM_STR, 25);

$sql->bindParam(':porcentaje_bateria',$porcentaje_bateria,PDO::PARAM_STR, 25);

//SE EJECUTA LA CONSULTA

$sql->execute();

?>
```

5.7. Diseño de la página web. Wordpress

La página web para mostrar los datos obtenidos con el dispositivo se ha creado con wordpress. Wordpress es un sistema de gestión de contenidos o CMS por sus siglas en inglés, Content Management System, diseñado para crear páginas web de cualquier tipo.

Wordpress funciona de manera muy intuitiva y permite crear y diseñar páginas web desde su panel de cotrol, sin necesidad de programar mediante código. Esta herramienta funciona mediante *plugins.* Los *plugins* son aplicaciones informáticas que se relacionan con otras para agregarles funciones específicas. Existen *plugins* de todo tipo: para gestionar bases de datos, para visualizar datos, para establecer la apariencia de la web, para venta online, sistemas de seguridad y demás.

Además, wordpress permite escoger entre infinidad de temas para seleccionar la apariencia deseada para tu página web.

5.7.1. Instalación de wordpress

En primer lugar, se descarga wordpress como un archivo zip desde la página oficial de la compañía y se descomprime en la raíz del almacenamiento de servidor web donde se aloja la página. A continuación, se accede a la web desde cualquier buscador. En la web aparecerá un panel de administración para instalar wordpress y asignarlo al dominio.

Durante este proceso de instalación se debe asignar la base de datos para el almacenamiento de los datos propios del wordpress y los plugins instalados. Esta base de datos no debe ser necesariamente la misma que se utiliza para almacenar los datos enviados por el dispositivo, pero por comodidad así se hizo. Para ello, durante la instalación de wordpress, se debe asignar el usuario administrador de la base de datos del hosting.

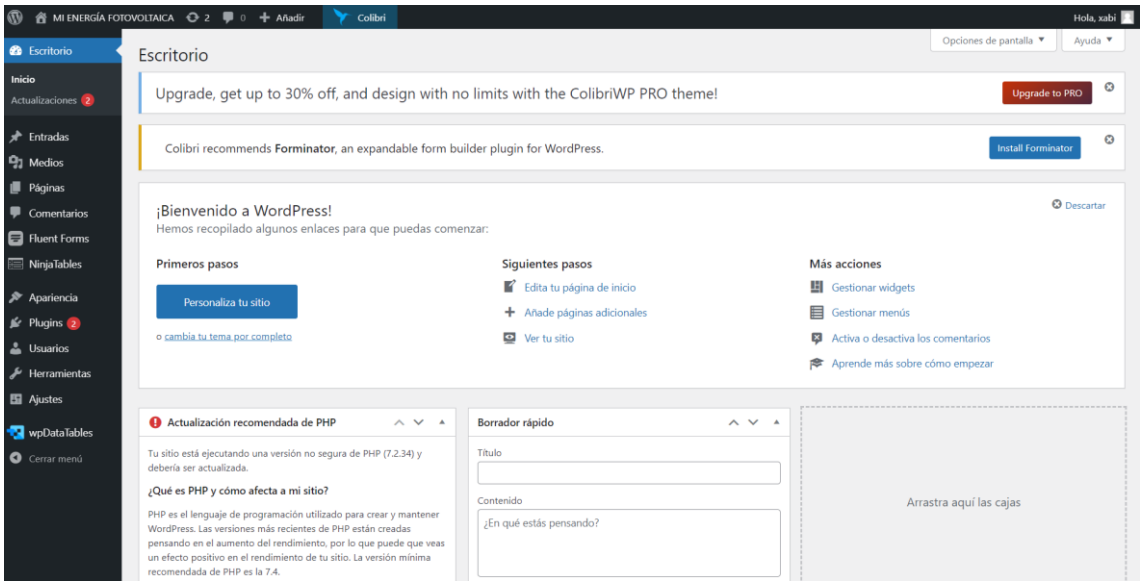


Fig 42 Panel de administración de Wordpress

5.7.2. Colibri Page Builder

El plugin escogido para la apariencia de la web es Colibri Page Builder. Como todos los *plugin,* Colibri Page Builder se instala desde el menú "Plugins" del panel de gestión de Wordpress.

Una vez instalado, Colibri permite crear páginas, escoger el tipo de letras, crear cuadros de texto, todo tipo de menús interactivos, introducir bloques y formas, entro muchas otras funciones. Gracias a esta herramienta, se ha escogido la siguiente apariencia para la página web.

Fig 43 Página principal de www.mienergiafotovoltaica.com

5.7.3. WpDataTables

El objetivo principal de la creació de esta página web es la representación gráfica de los datos obtenidos con el dispositivo. Para esta representación, se ha optado por un plugin muy potente llamado WpDataTables.

Esta herramienta permite crear tablas y gráficos interactivos de todo tipo. Además, permite utilizar los gráficos de High Charts y Google Charts y Charts.js.

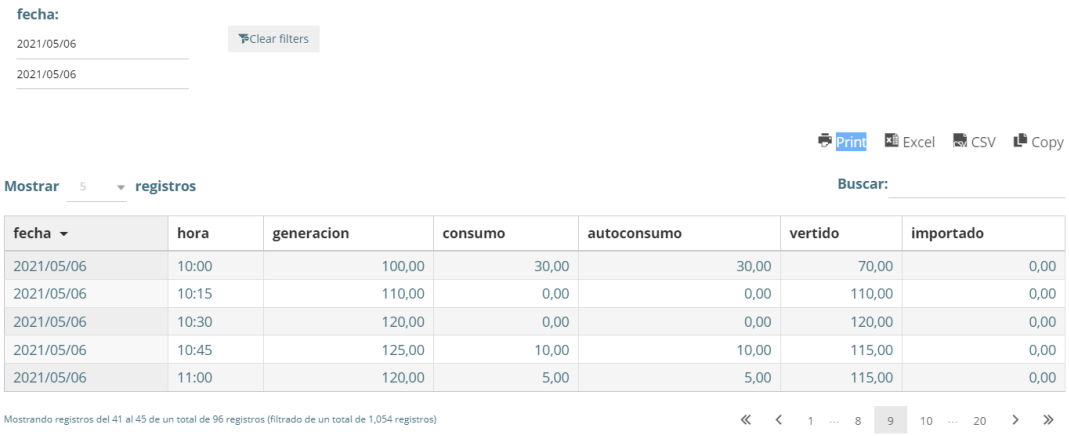
- Creación de tablas y gráficos con WpDataTables

En primer lugar, se crea una tabla que albergará los datos que se quieran graficar. Para ello, se genera una consulta sql a la base de datos donde se almacenan. Toda esta configuración se realiza desde el panel de gestión del plugin, sin necesidad de programar mediante código.

Una vez creada la tabla, se crea un gráfico para representar dichos datos. Se escoge el tipo de gráfico, se enlaza con una tabla de datos, se escogen los datos y su rango para cada eje y se configura la apariencia.

Una de las funciones más interesantes que ofrece este *plugin* es la creación de gráficos que respetan el filtrado de la tabla donde se encuentran sus datos. De esta manera, con un mismo gráfico se pueden representar los datos del día o los días deseados. Para activar esta función, se escoge la opción "Follow Table Filtering" al escoger el rango de datos a graficar.

Para la representación de los datos de generación, consumo, autoconsumo, excedentes e importación se ha utilizado la siguiente tabla.


fecha:
2021/05/06
2021/05/06
Clear filters
Print Excel CSV Copy
Mostrar 5 registros
Buscar:

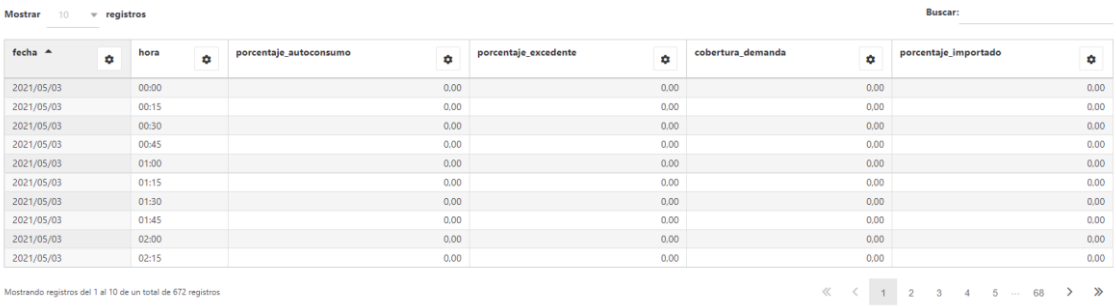
fecha	hora	generacion	consumo	autoconsumo	vertido	importado
2021/05/06	10:00	100,00	30,00	30,00	70,00	0,00
2021/05/06	10:15	110,00	0,00	0,00	110,00	0,00
2021/05/06	10:30	120,00	0,00	0,00	120,00	0,00
2021/05/06	10:45	125,00	10,00	10,00	115,00	0,00
2021/05/06	11:00	120,00	5,00	5,00	115,00	0,00

Mostrando registros del 41 al 45 de un total de 96 registros (filtrado de un total de 1,054 registros)
« ‹ 1 … 8 9 10 … 20 › »

Fig 44 Tabla de resultados

Esta tabla se podrá filtrar por días para que el usuario pueda obtener los datos del día que quiera de manera sencilla.

Los datos históricos se almacenan en la tabla siguiente:


Mostrar 10 registros
Buscar:

fecha	hora	porcentaje_autoconsumo	porcentaje_excedente	cobertura_demanda	porcentaje_importado
2021/05/03	00:00	0,00	0,00	0,00	0,00
2021/05/03	00:15	0,00	0,00	0,00	0,00
2021/05/03	00:30	0,00	0,00	0,00	0,00
2021/05/03	00:45	0,00	0,00	0,00	0,00
2021/05/03	01:00	0,00	0,00	0,00	0,00
2021/05/03	01:15	0,00	0,00	0,00	0,00
2021/05/03	01:30	0,00	0,00	0,00	0,00
2021/05/03	01:45	0,00	0,00	0,00	0,00
2021/05/03	02:00	0,00	0,00	0,00	0,00
2021/05/03	02:15	0,00	0,00	0,00	0,00

Mostrando registros del 1 al 10 de un total de 672 registros
« ‹ 1 2 3 4 5 … 68 › »

Fig 45 Tabla de datos históricos

Se han diseñado los siguientes gráficos con el fin de representar toda la información de interés para el usuario. Se muestran los datos de un día genérico.

- Evolución diaria de las variables por horas.

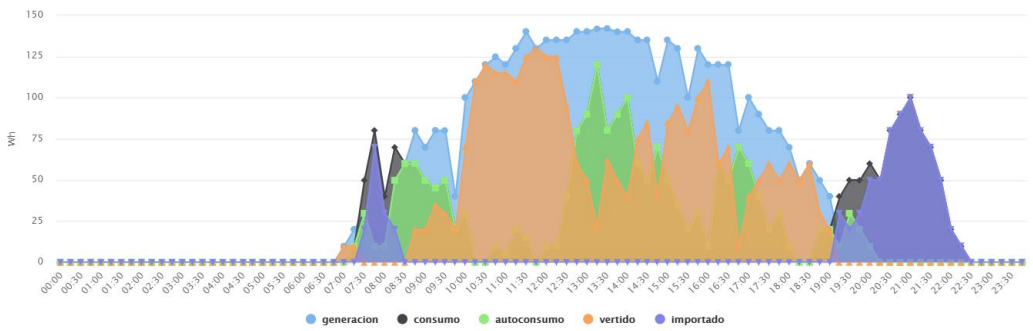


Fig 46 Gráfico de evolución diaria

Este gráfico es interactivo, cuando se posiciona el cursor sobe una de las áreas esta se destaca frente a las demás.

- Totales por días.

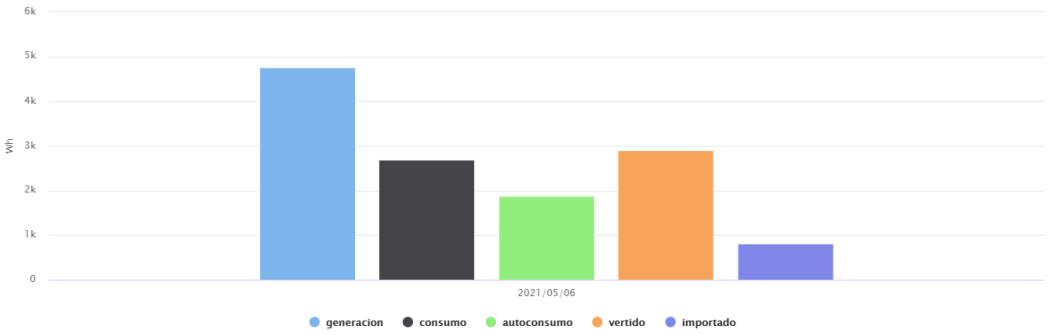


Fig 47 Gráfico de totales diarios

- Generación por horas.

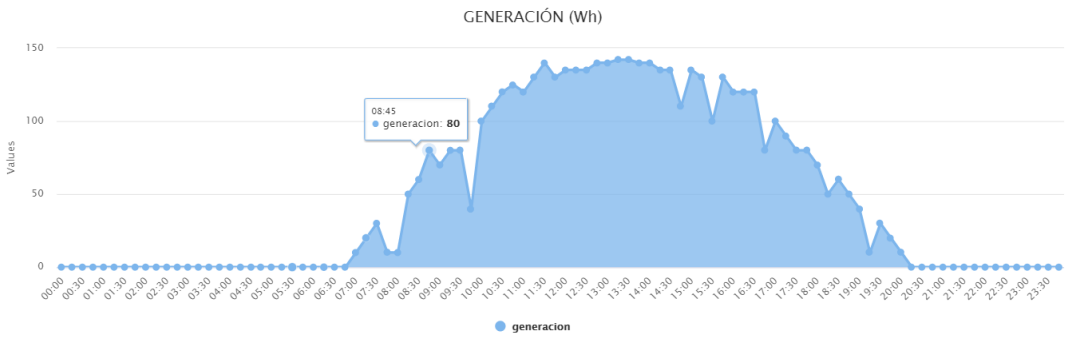


Fig 48 Gráfico de generación por horas

- Consumo por horas.

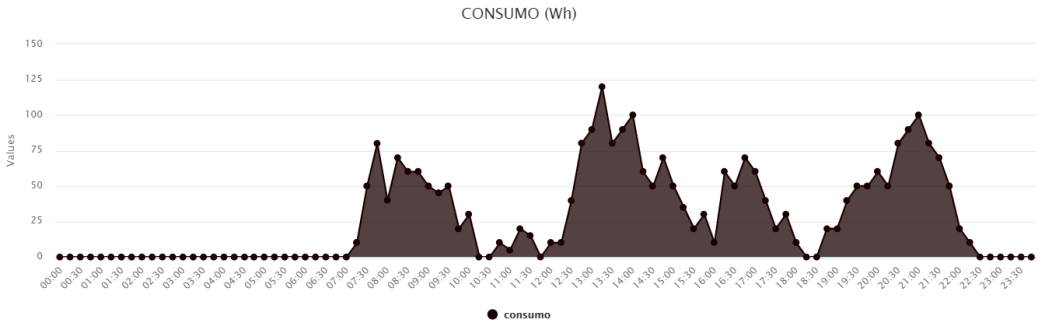


Fig 49 Gráfico de consumo por horas

- Autoconsumo por horas.

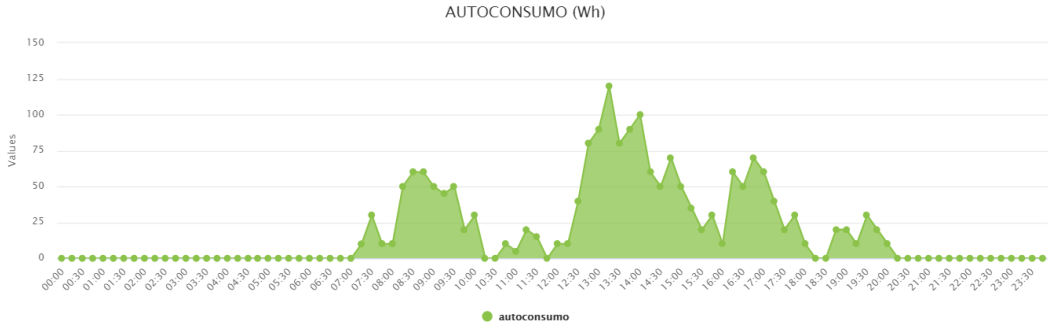


Fig 50 Gráfico de autoconsumo por horas

- Vertidos por horas.

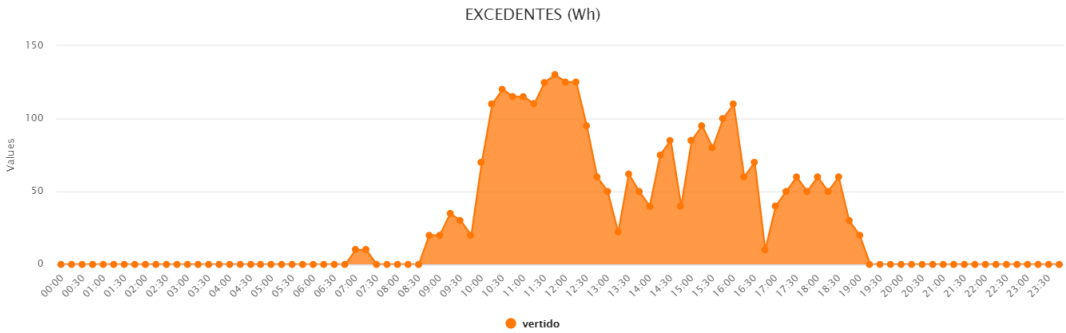


Fig 51 Gráfico de vertidos por horas

- Importación por horas.

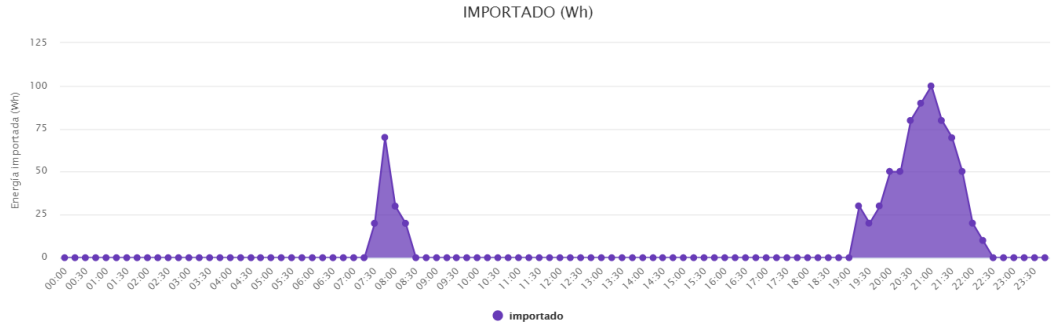


Fig 52 Gráfico de importación por horas

- Evolución histórica del porcentaje de autoconsumo.

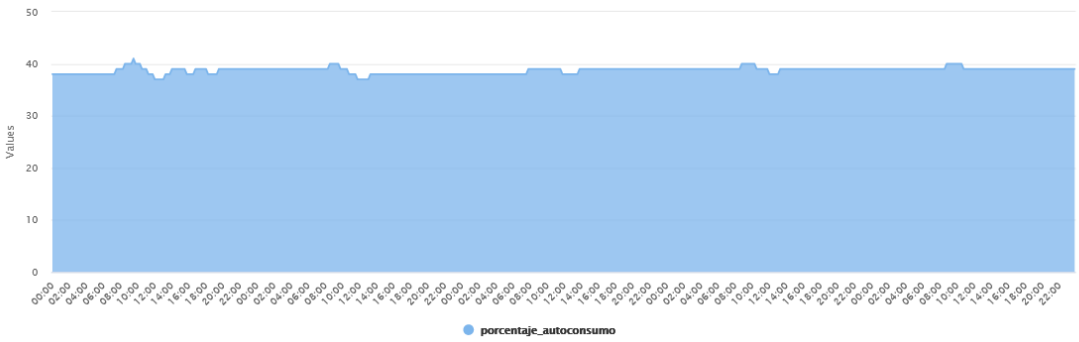


Fig 53 Gráfico de la evolución histórica del porcentaje de autoconsumo

- Evolución histórica de la cobertura de la demanda.

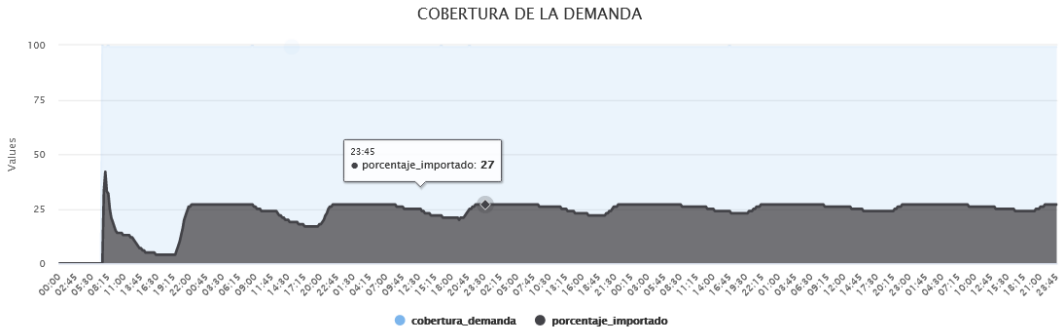


Fig 54 Gráfico de la evolución histórica de la cobertura de la demanda

BIBLIOGRAFÍA

[1] Arduino, [En línea]. Available: https://www.arduino.cc/. [Último acceso: 2021].

[2] «WordPress Official Site,» Wordpress, [En línea]. Available: https://wordpress.com/es/. [Último acceso: 2021].

[3] G. G. Alonso, «Medidor de consumo,» 2015. [En línea]. Available: http://www.gonzalogalvan.es/category/desarrollos/medidor-de-consumo/. [Último acceso: 2021].

[4] L. Llamas, «Entrada analógica de 16 bit con Arduino y ADS1115,» 14 11 2016. [En línea]. Available: https://www.luisllamas.es/entrada-analogica-adc-de-16-bits-con-arduino-y-ads1115/. [Último acceso: 2021].

[5] Arduino, «Code: Web Client,» [En línea]. Available: https://playground.arduino.cc/Code/WebClient/. [Último acceso: 2021].

[6] R. Ingenieril, «Enviar datos de Arduino a MySQL con ethernet,» [En línea]. Available: https://www.rinconingenieril.es/arduino-ethernet-y-mysql/ . [Último acceso: 2021].

[7] B. PHP, «Insertar datos MySQL usando PDO PHP,» [En línea]. Available: https://www.baulphp.com/insertar-datos-mysql-usando-pdo-php/#Insertar_datos_MySQL_usando_PDO_PHP. [Último acceso: 2021].

[8] WebTemática, «Como hacer una página web en WordPress,» [En línea]. Available: https://webtematica.com/diseno-web-wordpress/como-hacer-una-pagina-web-en-wordpress.

[9] D. web, «Tipos de datos en MySQL,» 2003. [En línea]. Available: https://desarrolloweb.com/articulos/1054.php#fecha . [Último acceso: 2021].

[10] Biblioteca de la Universidad Pública de Navarra. Oficina de Referencia, «Guía para citar y referenciar. IEEE Style,» 2016. [En línea]. Available: https://goo.gl/LaUj46.. [Último acceso: 2021].

ANEXOS

ANEXO I: Programación para autoconsumo trifásico con baterías

```
1. //////// AUTOCONSUMO CON BATERÍAS TRIFÁSICO /////////
2. ////////////////////////////////////////////////////
3.
4. //CONEXIONES ADC INTERNO ARDUINO
5. //A0 --> - del shunt
6. //A1 --> + del shunt
7. //A2 --> pin IAC del inversor (FASE R)
8. //A3 --> pin IAC del inversor (FASE S)
9. //A4 --> pin IAC del inversor (FASE T)
10.      //A5 --> pin IAC de red (FASE R)
11.      //A6 --> pin IAC de red (FASE S)
12.      //A7 --> pin IAC de red (FASE T)
13.      //A8 --> pin VAC inversor (FASE R)
14.      //A9 --> pin VAC inversor (FASE S)
15.      //A10 --> pin VAC inversor (FASE T)
16.      //A11 --> pin VAC de red (FASE R)
17.      //A12 --> pin VAC de red (FASE S)
18.      //A13 --> pin VAC de red (FASE T)
19.
20.      //CONEXIONES ADS1115
21.      //VDD --> 5V
22.      //GND --> GND
23.      //SCL --> SCL PLACA
24.      //SDA --> SDA PLACA
25.      //ADDR --> GND (Dirección por defecto)
26.      //ALRT --> SIN CONECTAR
27.      //A0 --> - DEL DIVISOR DE TENSIÓN
28.      //A1 --> + DEL DIVISOR DE TENSIÓN
29.
30.      //CONEXIONES PINES DIGITALES
31.      //2 --> Vout DEL SENSOR DE Tª
32.      //50 --> SEÑAL DE SALIDA DEL RELÉ (FASE R)
33.      //51 --> SEÑAL DE SALIDA DEL RELÉ (FASE S)
34.      //52 --> SEÑAL DE SALIDA DEL RELÉ (FASE T)
35.
36.
37.      //LIBRERÍAS NECESARIAS PARA EL ADS1115 Y LA COMUNICACIÓN
   I20
38.      #include <Adafruit_BusIO_Register.h>
39.      #include <Adafruit_I2CDevice.h>
40.      #include <Adafruit_I2CRegister.h>
41.      #include <Adafruit_SPIDevice.h>
42.      #include <Adafruit_ADS1X15.h>
43.      #include <Wire.h>
44.
45.      //Se define la instancia para un ADC
46.      Adafruit_ADS1115 adc;
47.      const float ganancia = 0.1875F; //Depende de la ganancia
   escogida para cada ADC
48.      //(GAIN_TWOTHIRDS, 6.144V de ref, 0.1875 mV por bit)
49.
50.      //LIBRERÍAS NECESARIAS PARA EL ENVIO DE DATOS CON ETHERNET
   AL SERVIDOR
51.      #include <SPI.h>
52.      #include <Ethernet.h>
53.
```

```
54.      EthernetClient client;
55.
56.      byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
57.      char serverName[] = "www.mienergiafotovoltaica.com";     //N
   ombre del dominio propio
58.      int serverPort = 80;     //Puerto para la transmisión de
   datos
59.      char pageAdd[64];
60.      int totalCount = 0;
61.      int FrecuenciaEnvio= 300000; //5 min = 300000 ms
62.
63.
64.      char Rele;
65.
66.
67.
68.      //GANANCIA DEL DIVISOR DE TENSIÓN PARA EL VOLTAJE DC (Medir
   valores reales de las resistencias)
69.      const float R1= 14740;
70.      const float R2= 3012;
71.      const float G= R2/(R1+R2);
72.
73.
74.      //LIBRERÍA PARA CALCULAR LA CORRIENTE RMS
75.      #include <EmonLib.h>
76.      //Se crean instancias para la librería (una para cada
   sensor de corriente)
77.      EnergyMonitor inv1;     //Se crea una instancia para la
   librería --> Sensor de corriente de salida del inversor
78.      EnergyMonitor inv2;     //Se crea una instancia para la
   librería --> Sensor de corriente de salida del inversor
79.      EnergyMonitor inv3;     //Se crea una instancia para la
   librería --> Sensor de corriente de salida del inversor
80.      EnergyMonitor red1;     //Se crea una instancia para la
   librería --> Sensor de corriente de red
81.      EnergyMonitor red2;     //Se crea una instancia para la
   librería --> Sensor de corriente de red
82.      EnergyMonitor red3;     //Se crea una instancia para la
   librería --> Sensor de corriente de red
83.
84.      //////////CALIBRACIÓN DE LOS SENSORES DE VOLTAJE AC
85.      //Los valores de vol_multi, adc_max y adc_min se obtienen
   de la calibración del sensor de voltaje AC
86.      //Reemplazar los valores adc_max, adc_min y volt_multi por
   los entregados por el sketch: volt_ac_cal para cada sensor
87.
88.      //Sensores de voltaje fase-neutro a la salida del inversor
89.      int adc_max_inv1=700;
90.      int adc_min_inv1=322;
91.      float volt_multi_inv1=230.1;
92.      float volt_multi_p_inv1;
93.      float volt_multi_n_inv1;
94.
95.      int adc_max_inv2=700;
96.      int adc_min_inv2=322;
97.      float volt_multi_inv2=230.1;
98.      float volt_multi_p_inv2;
99.      float volt_multi_n_inv2;
100.
101.     int adc_max_inv3=700;
102.     int adc_min_inv3=322;
```

```
103.      float volt_multi_inv3=230.1;
104.      float volt_multi_p_inv3;
105.      float volt_multi_n_inv3;
106.
107.      //Sensores de voltaje fase-neutro a la salida de la
   acometida
108.      int adc_max_red1=700;
109.      int adc_min_red1=322;
110.      float volt_multi_red1=230.1;
111.      float volt_multi_p_red1;
112.      float volt_multi_n_red1;
113.
114.      int adc_max_red2=700;
115.      int adc_min_red2=322;
116.      float volt multi red2=230.1;
117.      float volt_multi_p_red2;
118.      float volt_multi_n_red2;
119.
120.      int adc_max_red3=700;
121.      int adc_min_red3=322;
122.      float volt_multi_red3=230.1;
123.      float volt_multi_p_red3;
124.      float volt_multi_n_red3;
125.
126.      //VARIABLES PARA EL CÁLCULO DE POTENCIA Y ENERGÍA
127.      float Pgen, Egen;
128.      float Pimp, Pex, Pcons, Econs, Eex, Eimp;
129.      float Pautoc, Eautoc;
130.      float GENERACION, CONSUMO,
   AUTOCONSUMO;              //Almacenan los valores totales
   históricos para calcular el porcentaje de autoconsumo y la
   cobertura de la demanda
131.      float porcentaje_autoconsumo, porcentaje_excedente,
   cobertura_demanda, porcentaje_importado;
132.      float porcentaje_bateria;
133.      float Ebat;
134.
135.      //SE DEFINEN LAS VARIABLES DEL TEMPORIZADOR
136.      long tiempo = 0;
137.      long tiempototal, tiempoproceso;
138.
139.      extern volatile unsigned long timer0_millis;
140.      unsigned long nuevo_valor = 0;
141.      int N;
142.
143.      //CAPACIDAD DE LAS BATERÍAS EN Wh
144.      float cargamax = 2000;
145.
146.      //PINES DE CONEXIÓN PARA LOS RELÉS DE GESTIÓN DE EXCEDENTES
147.      int rele_r = 50;
148.      int rele_s = 51;
149.      int rele_t = 52;
150.
151.
152.      void setup() {
153.
154.        Serial.begin(115200);
155.
156.      //Se desabilita la comunicación SPI con la SD del módulo
   ethernet
157.        pinMode(4,OUTPUT);
```

```
158.       digitalWrite(4,HIGH);
159.
160.       Serial.print(F("Starting ethernet..."));
161.       if(!Ethernet.begin(mac)) Serial.println(F("failed"));
162.       else Serial.println(Ethernet.localIP());
163.
164.       delay(2000);
165.       Serial.println(F("Ready"));
166.
167.     //CORRIENTE AC
168.     //IMPORTANTE: Calibrar correctamente la pinza amperimétrica
   (35 para la pinza de 30A)
169.       inv1.current(A2, 35);                    // Current:
   input pin, calibration.
170.       inv2.current(A3, 35);                    // Current:
   input pin, calibration.
171.       inv3.current(A4, 35);                    // Current:
   input pin, calibration.
172.       red1.current(A5, 35);                    // Current:
   input pin, calibration.
173.       red2.current(A6, 35);                    // Current:
   input pin, calibration.
174.       red3.current(A7, 35);                    // Current:
   input pin, calibration.
175.
176.
177.     //VOLTAJE AC
178.       volt_multi_p_inv1 = volt_multi_inv1 * 1.4142;   //Voltaje
   pico= Voltaje RMS * 1.4142
179.       volt_multi_n_inv1 = volt_multi_p_inv1 * -1;
180.
181.       volt_multi_p_inv2 = volt_multi_inv2 * 1.4142;   //Voltaje
   pico= Voltaje RMS * 1.4142
182.       volt_multi_n_inv2 = volt_multi_p_inv2 * -1;
183.
184.       volt_multi_p_inv3 = volt_multi_inv3 * 1.4142;   //Voltaje
   pico= Voltaje RMS * 1.4142
185.       volt_multi_n_inv3 = volt_multi_p_inv3 * -1;
186.
187.       volt_multi_p_red1 = volt_multi_red1 * 1.4142;   //Voltaje
   pico= Voltaje RMS * 1.4142
188.       volt_multi_n_red1 = volt_multi_p_red1 * -1;
189.
190.       volt_multi_p_red2 = volt_multi_red2 * 1.4142;   //Voltaje
   pico= Voltaje RMS * 1.4142
191.       volt_multi_n_red2 = volt_multi_p_red2 * -1;
192.
193.       volt_multi_p_red3 = volt_multi_red3 * 1.4142;   //Voltaje
   pico= Voltaje RMS * 1.4142
194.       volt_multi_n_red3 = volt_multi_p_red3 * -1;
195.
196.     //VOLTAJE DC
197.     //Configuración del adc externo ADS1115
198.       adc.begin();                        // Inicializar el adc4
   con la dirección por defecto --> pin addr conectado a GND
199.       adc.setGain(GAIN_TWOTHIRDS);        // +/- 6.144V de
   referencia --> 1 bit = 0.1875mV
200.
201.     //INICIO DE LOS CONTADORES PARA CALCULO DE POTENCIAS
202.       tiempototal= 0;                     // Tiempo cuando se
   empieza a ejecutar cada loop
```

```
203.       tiempo = 0;                          // Tiempo desde que
   se empezó a ejecutar
204.       N = 0;
205.
206.     //INICIO DE ACUMULACIÓN DE DATOS HISTÓRICOS
207.     //IMPORTANTE: MODIFICAR ESTOS VALORES SI SE RESETEA CÓDIGO
   Y NO SE QUIERE REINICIAR LA CUENTA
208.
209.       GENERACION = 0;
210.       CONSUMO = 0;
211.       AUTOCONSUMO = 0;
212.
213.       pinMode(rele_r, OUTPUT);
214.       pinMode(rele_s, OUTPUT);
215.       pinMode(rele t, OUTPUT);
216.
217.     //Valor inicial de la carga de las baterías
218.       Ebat = cargamax;
219.     }
220.
221.
222.
223.     void loop() {
224.
225.     //DECLARACIÓN DE LOS CONTADORES
226.      tiempototal = tiempo;
227.      tiempo = millis();
228.
229.
230.     //////////MEDIDAS AC: SALIDA DEL INVERSOR Y DE LA
   ACOMETIDA//////////
231.     //VOLTAJE AC
232.
233.       float Vinv1, Vinv2, Vinv3, Vred1, Vred2,
   Vred3 =get_voltage(); //Voltage eficaz (V-RMS)
234.
235.     //DESEQUILIBRIO DE FASES
236.     //Desequilibrio a la salida del inversor
237.     float Vmedia = (Vinv1 + Vinv2 + Vinv3)/3;
238.     float desequilibrio;
239.
240.       desequilibrio = (max(max(abs(Vinv1-Vinv2),abs(Vinv1-
   Vinv3)),abs(Vinv2-Vinv3))/Vmedia)*100;
241.
242.     //Desequilibrio a la salida del inversor
243.     float Vmedia_red = (Vred1 + Vred2 + Vred3)/3;
244.     float desequilibrio;
245.
246.       desequilibrio_red = (max(max(abs(Vred1-Vred2),abs(Vred1-
   Vred3)),abs(Vred2-Vred3))/Vmedia_red)*100;
247.
248.       if (desequilibrio > 2){
249.         digitalWrite(11, HIGH); //Enciende el led de alarma
250.       }else{
251.         digitalWrite(11, LOW);
252.       }
253.       if (desequilibrio > 2){
254.         digitalWrite(12, HIGH); //Enciende el led de alarma
255.       }else{
256.         digitalWrite(12, LOW);
257.       }
```

```
258.
259.        Serial.println("VOLTAJES DE FASE A LA SALIDA DEL
   INVERSOR");
260.        Serial.print("Fase R: ");
261.        Serial.print(Vinv1,3);
262.        Serial.println(" V");
263.        Serial.print("Fase S: ");
264.        Serial.print(Vinv2,3);
265.        Serial.println(" V");
266.        Serial.print("Fase T: ");
267.        Serial.print(Vinv3,3);
268.        Serial.println(" V");
269.
270.        Serial.println("VOLTAJES DE FASE A LA SALIDA DE LA
   ACOMETIDA");
271.        Serial.print("Fase R: ");
272.        Serial.print(Vred1,3);
273.        Serial.println(" V");
274.        Serial.print("Fase s: ");
275.        Serial.print(Vred2,3);
276.        Serial.println(" V");
277.        Serial.print("Fase t: ");
278.        Serial.print(Vred3,3);
279.        Serial.println(" V");
280.
281.     //CORRIENTE AC
282.     //Corrientes a la salida del inversor
283.       double Iinv1= inv1.calcIrms(1480);
284.       double Iinv2= inv2.calcIrms(1480);
285.       double Iinv3= inv3.calcIrms(1480);
286.
287.     //Corrientes a la salida de la acometida
288.       double Ired1 = red1.calcIrms(1480);
289.       double Ired2 = red2.calcIrms(1480);
290.       double Ired3 = red3.calcIrms(1480);
291.
292.        Serial.println("CORRIENTES DE FASE A LA SALIDA DEL
   INVERSOR");
293.        Serial.print("Fase R: ");
294.        Serial.print(Iinv1,3);
295.        Serial.println(" A");
296.        Serial.print("Fase S: ");
297.        Serial.print(Iinv2,3);
298.        Serial.println(" A");
299.        Serial.print("Fase T: ");
300.        Serial.print(Iinv3,3);
301.        Serial.println(" A");
302.
303.        Serial.println("CORRIENTES DE FASE A LA SALIDA DE LA
   ACOMETIDA");
304.        Serial.print("Fase R: ");
305.        Serial.print(Ired1,3);
306.        Serial.println(" A");
307.        Serial.print("Fase S: ");
308.        Serial.print(Ired2,3);
309.        Serial.println(" A");
310.        Serial.print("Fase T: ");
311.        Serial.print(Ired3,3);
312.        Serial.println(" A");
313.
314.     //////////MEDIDAS DC: BATERÍAS//////////
```

```
315.      //VOLTAJE DC
316.      int16_t lectura;
317.      float voltaje;
318.      float V, V_bat;
319.      int contador= 0;
320.      V_bat= 0;
321.
322.        for (int i = 0; i < 100; i ++) //toma 100 medidas por
   cada segundo y luego se calcula la media
323.        {
324.          lectura = adc.readADC_Differential_0_1();  //lectura
   del adc en modo diferencial entre A1 y A0
325.          voltaje = lectura*ganancia/1000;  //Valor de la lectura
   en voltios (escala 0-5V)
326.          V = voltaje/G;  //Voltaje real de la batería (5V -->
   30V)
327.          V_bat = V_bat + V;
328.          contador = contador +1;
329.          delay(10);
330.        }
331.
332.        V_bat = V_bat/contador;
333.        Serial.print("Vdc= ");
334.        Serial.print(V_bat);
335.        Serial.println(" V");
336.
337.
338.      //CORRIENTE DC
339.        analogReference(INTERNAL1V1);    //Referencia interna del
   arduino Mega de 1,1V
340.        delay(10);                       //Para estabilizar la
   placa después del cambio de referencia
341.
342.      //Lectura de la corriente que circula por el shunt
343.        int valor;
344.        float corriente;
345.        float shunt;
346.        float Idc=0;
347.        int cont = 0;
348.
349.        for (int i = 0; i < 100; i ++) //toma 100 medidas por
   cada segundo y luego se calcula la media
350.      {
351.        valor= analogRead(A0)-analogRead(A1);    //Diferencia
   entre los dos bornes del shunt en bits
352.        shunt = (lectura*1.1/1024)*1000;          //1,1V son 1024
   bits
353.        corriente = shunt*100/75;               //100 A son 75mV
354.        Idc = Idc + corriente;
355.        cont = cont + 1;
356.        delay(10);
357.      }
358.        Idc = Idc/cont;
359.
360.        Serial.print("Idc= ");
361.        Serial.print(Idc);
362.        Serial.println(" A");
363.        analogReference(DEFAULT);             //Se reestablece al
   referencia de 5V para el adc interno
364.
365.
```

```
366.     ////////////////////////////////////////////////////////////////
   ////////////////////////////
367.     ////CÁLCULOS
368.       tiempoproceso= tiempo-tiempototal;    //Se calcula el
   tiempo que ha pasado desde la última vez que se ejecutó el loop
369.       Serial.print("Tiempo proceso: ");
370.       Serial.print(tiempoproceso);
371.       Serial.println(" ms");
372.
373.     float T= tiempoproceso/1000;            //Tiempo del loop
   en segundos
374.
375.     //Potencia de las baterías
376.     float Pbat, Ebat;
377.
378.       Pbat= V_bat * Idc;                    //>0 si es de carga
379.
380.     //Potencia de salida del inversor
381.     float Pinv, Einv;
382.
383.       Pinv = Vinv1*Iinv1 + Vinv2*Iinv2 + Vinv3*Iinv3;
            //Se supone FP=1
384.
385.     //Potencia intercambiada con la red
386.     float Pred, Ered;
387.
388.       Pred = Vred1*Ired1 + Vred2*Ired2 + Vred3*Ired3;
            //Se supone FP=1
389.
390.     //GENERACIÓN
391.     //Cuando la potencia en las baterías es positiva (carga) o
   nula es potencia generada --> Pgen = Pinv + Pbat
392.     //Cuando la potencia en las baterías en negativa (descarga)
   no es potencia generada --> Pgen = Pinv - Pbat --> Como en ese
   caso la Pbat es <0 --> P gen = Pinv + Pbat
393.
394.      Pgen= Pinv + Pbat;
395.
396.      Egen = Egen + Pgen*T/3600;
397.
398.     //IMPORTACIÓN, EXCEDENTES y CONSUMO
399.     //Si la potencia de la red es positiva, se está exportando
   energía (por criterio de signos)
400.     //Si se está importando energía y a la vez generando, el
   consumo total es la suma de ambos conceptos
401.
402.       if (Pred >= 0){
403.         Pex = Pred;
404.         Pimp = 0;
405.       }
406.       else {
407.         Pimp = -Pred;     //Potencia importada en valor
   absoluto
408.         Pex = 0;
409.       }
410.       Pcons = Pgen - Pex + Pimp;
411.       Eex = Eex + Pex*T/3600;
412.       Eimp = Eimp + Pimp*T/3600;
413.       Econs = Econs + Pcons*T/3600;
414.
415.     //AUTOCONSUMO
```

```
416.     //Cuando hay excedentes, la energía autoconsumida es la
   resta entre la generación y los excedentes
417.     //Cuando se importa energía la energía autoconsumida es
   toda la generada
418.
419.       if (Pex >0){
420.         Pautoc = Pgen - Pex;
421.       }
422.       else {
423.         Pautoc = Pgen;
424.       }
425.       Eautoc = Eautoc + Pautoc*T/3600;
426.
427.     //PORCENTAJE DE AUTOCONSUMO ACUMULADO
428.     //Generación total y consumo total acumulados
429.       GENERACION= GENERACION + Egen;
430.       CONSUMO = CONSUMO + Econs;
431.       AUTOCONSUMO = AUTOCONSUMO + Eautoc;
432.
433.       porcentaje_autoconsumo = (AUTOCONSUMO/GENERACION)*100;
434.       porcentaje_excedente = 100 - porcentaje_autoconsumo;
435.
436.       cobertura_demanda = (AUTOCONSUMO/CONSUMO)*100;
437.       porcentaje_importado = 100 - cobertura_demanda;
438.
439.
440.
441.     //ESTADO DE CARGA DE LAS BATERÍAS
442.     //Se calcula el nivel de carga de las baterías en función
   de la energía almacenada
443.
444.       Ebat = Ebat + Pbat*T/3600;
445.
446.       if (Ebat > cargamax) {
447.         Ebat = cargamax;}
448.
449.       porcentaje_bateria = (Ebat/cargamax)*100;
450.
451.       if (porcentaje_bateria > 80){
452.         digitalWrite(carga20, HIGH);
453.         digitalWrite(carga40, HIGH);
454.         digitalWrite(carga60, HIGH);
455.         digitalWrite(carga80, HIGH);
456.         digitalWrite(carga100, HIGH);
457.         }
458.       else if (80 >= porcentaje_bateria > 60){
459.         digitalWrite(carga20, HIGH);
460.         digitalWrite(carga40, HIGH);
461.         digitalWrite(carga60, HIGH);
462.         digitalWrite(carga80, HIGH);
463.         digitalWrite(carga100, LOW);
464.       } else if (60 >= porcentaje_bateria > 40){
465.         digitalWrite(carga20, HIGH);
466.         digitalWrite(carga40, HIGH);
467.         digitalWrite(carga60, HIGH);
468.         digitalWrite(carga80, LOW);
469.         digitalWrite(carga100, LOW);
470.       }
471.
472.         else if (40 >= porcentaje_bateria > 20){
473.         digitalWrite(carga20, HIGH);
```

```
474.          digitalWrite(carga40, HIGH);
475.          digitalWrite(carga60, LOW);
476.          digitalWrite(carga80, LOW);
477.          digitalWrite(carga100, LOW);
478.        }
479.          else if (20 >= porcentaje_bateria ){
480.          digitalWrite(carga20, HIGH);
481.          digitalWrite(carga40, LOW);
482.          digitalWrite(carga60, LOW);
483.          digitalWrite(carga80, LOW);
484.          digitalWrite(carga100, LOW);
485.        }
486.        else if (porcentaje_bateria = 0){
487.          digitalWrite(carga20, LOW);
488.          digitalWrite(carga40, LOW);
489.          digitalWrite(carga60, LOW);
490.          digitalWrite(carga80, LOW);
491.          digitalWrite(carga100, LOW);
492.        }
493.
494.      //GESTIÓN DE EXCEDENTES
495.        if (Pex >0){
496.          //Cuando hay excedentes se activa la salida del relé
497.          digitalWrite(rele_r, HIGH);
498.          digitalWrite(rele_s, HIGH);
499.          digitalWrite(rele_t, HIGH);
500.          digitalWrite(13, HIGH);
501.          }
502.        else {
503.          digitalWrite(rele_r, LOW);
504.          digitalWrite(rele_s, LOW);
505.          digitalWrite(rele_t, LOW);
506.        }
507.
508.       if (tiempototal > FrecuenciaEnvio){
509.        //ENVÍO DE LOS DATOS ALMACENADOS EN LOS ÚLTIMOS 5 MIN
510.        //Se envía Egen, Econs, Eimp, Eex, Eautoc,
   porcentaje_autoconsumo,
511.        //porcentaje_excedente, cobertura_demanda,
   porcentaje_importado y porcentaje_bateria
512.
513.          sprintf(pageAdd,"/databaseconnect.php?lectura=%s",total
   Count);
514.
515.        //SI FALLA LA CONEXIÓN AVISA, SI NO SE LLAMA getPage
516.          if(!getPage(serverName,serverPort,pageAdd)) Serial.prin
   t(F("Fail "));
517.          else Serial.print(F("Pass "));
518.          totalCount++;
519.          Serial.println(totalCount,DEC);
520.
521.        //RESETEO DE LOS VALORES DE ENERGÍA ACUMULADOS
522.        Egen =0;
523.        Eimp =0;
524.        Eex =0;
525.        Econs =0;
526.        Eautoc =0;
527.
528.       }
529.
530.     }
```

```
531.
532.
533.
534.
535.
536.      //CÁLCULO DEL VOLTAJE AC RMS
537.      //IMPORTANTE: NECESITA 5V COMO REFERENCIA ANALÓGICA
538.      //DEVUELVE EL VOLTAJE DE SALIDA DEL INVERSOR Y EL VOLTAJE
   DE SALIDA DE LA ACOMETIDA
539.      float get_voltage(void)
540.      {
541.        //analogReference(DEFAULT); //POR SI SE HA CAMBIADO EN
   OTRO MOMENTO
542.        float adc_inv1, adc_inv2, adc_inv3, adc_red1, adc_red2,
   adc_red3;
543.        float volt_inst1=0;
544.        float volt_inst2=0;
545.        float volt_inst3=0;
546.        float volt_inst_red1=0;
547.        float volt_inst_red2=0;
548.        float volt_inst_red3=0;
549.        float Sum1=0;
550.        float Sum2=0;
551.        float Sum3=0;
552.        float Sum4=0;
553.        float Sum5=0;
554.        float Sum6=0;
555.        float V_inv1, V_inv2, V_inv3, V_red1, V_red2, V_red3;
556.        long tiempo_init=millis();
557.        int N=0;
558.
559.        while( (millis() - tiempo_init) < 600)//Duración 0.6
   segundos(Aprox. 30 ciclos de 50Hz)
560.        {
561.
562.          adc_inv1 = analogRead(A8);    //Entrada sensor de
   voltaje del inversor
563.          adc_inv2 = analogRead(A9);    //Entrada sensor de
   voltaje del inversor
564.          adc_inv3 = analogRead(A10);    //Entrada sensor de
   voltaje del inversor
565.
566.          adc_red1 = analogRead(A11);    //Entrada sensor de
   voltaje de la red
567.          adc_red2 = analogRead(A12);    //Entrada sensor de
   voltaje de la red
568.          adc_red3 = analogRead(A13);    //Entrada sensor de
   voltaje de la red
569.
570.          volt_inst1 = map(adc_inv1,adc_min_inv1,adc_max_inv1,vol
   t_multi_n_inv1,volt_multi_p_inv1);
571.          volt_inst2 = map(adc_inv2,adc_min_inv2,adc_max_inv2,vol
   t_multi_n_inv2,volt_multi_p_inv2);
572.          volt_inst3 = map(adc_inv3,adc_min_inv3,adc_max_inv3,vol
   t_multi_n_inv3,volt_multi_p_inv3);
573.
574.          volt_inst_red1 = map(adc_red1,adc_min_red1,adc_max_red1
   ,volt_multi_n_red1,volt_multi_p_red1);
575.          volt_inst_red2 = map(adc_red2,adc_min_red2,adc_max_red2
   ,volt_multi_n_red2,volt_multi_p_red2);
```

```
576.          volt_inst_red3 = map(adc_red3,adc_min_red3,adc_max_red3
   ,volt_multi_n_red3,volt_multi_p_red3);
577.
578.      //Sumatoria de Cuadrados
579.          Sum1 = Sum1+sq(volt_inst1);
580.          Sum2 = Sum2+sq(volt_inst2);
581.          Sum3 = Sum3+sq(volt_inst3);
582.
583.          Sum4 = Sum4+sq(volt_inst_red1);
584.          Sum5 = Sum5+sq(volt_inst_red2);
585.          Sum6 = Sum6+sq(volt_inst_red3);
586.
587.          N = N+1;
588.          delay(1);
589.        }
590.      //ecuación del RMS
591.        V_inv1=sqrt((Sum1)/N);
592.        V_inv2=sqrt((Sum2)/N);
593.        V_inv3=sqrt((Sum3)/N);
594.        V_red1=sqrt((Sum4)/N);
595.        V_red2=sqrt((Sum5)/N);
596.        V_red3=sqrt((Sum6)/N);
597.
598.        return V_inv1, V_inv2, V_inv3, V_red1, V_red2, V_red3;
599.      }
600.
601.      //RESETEO DEL CONTADOR para que no se desborde
602.      void setMillis(unsigned long nuevo_valor){
603.        uint8_t oldSREG = SREG;
604.        cli();
605.        timer0_millis = nuevo_valor;
606.        SREG = oldSREG;
607.      }
608.
609.
610.      //ENVÍO DE LOS DATOS
611.      byte getPage (char* domainBuffer,int thisPort,char* page)
612.      { int inChar;
613.        char outBuf[64];
614.        Serial.print(F("connecting..."));
615.        if(client.connect(domainBuffer,thisPort) == 1)
616.        { Serial.print(F("connected to "));
617.          Serial.println(domainBuffer);
618.          Serial.println();
619.          client.print("GET /databaseconnect.php?generacion=");
620.          client.print(Egen);
621.          client.print("&consumo=");
622.          client.print(Econs);
623.          client.print("&autoconsumo=");
624.          client.print(Eautoc);
625.          client.print("&exportado=");
626.          client.print(Eex);
627.          client.print("&importado=");
628.          client.print(Eimp);
629.          client.print("&porcentaje_autoconsumo=");
630.          client.print(porcentaje_autoconsumo);
631.          client.print("&porcentaje_excedente=");
632.          client.print(porcentaje_excedente);
633.          client.print("&cobertura_demanda=");
634.          client.print(cobertura_demanda);
635.          client.print("&porcentaje_importado=");
```

```
636.          client.print(porcentaje_importado);
637.          client.print("&porcentaje_bateria=");
638.          client.print(porcentaje_bateria);
639.          client.println(" HTTP/1.1");
640.
641.          sprintf(outBuf,"Host: %s",serverName);
642.          client.println(outBuf);
643.          client.println(F("Connection: close\r\n")); }
644.        else
645.        { Serial.println(F("failed"));
646.          return 0; }
647.        int connectLoop = 0;
648.        while(client.connected())
649.        {
650.          while(client.available())
651.          {
652.            inChar = client.read();
653.            Serial.write(inChar);
654.            connectLoop = 0; }
655.          delay(1);
656.          connectLoop++;
657.          if(connectLoop > 10000)
658.          {
659.            Serial.println();
660.            Serial.println(F("Timeout"));
661.            client.stop(); } }
662.        Serial.println();
663.        Serial.println(F("disconnecting."));
664.        client.stop();
665.        return 1;
666.      }
```

ANEXO II: Programación para autoconsumo monofásico con baterías

```
1. ////////// AUTOCONSUMO CON BATERÍAS MONOFÁSICO //////////
2. ////////////////////////////////////////////////////////
3.
4. //CONEXIONES ADC INTERNO ARDUINO
5. //A0 --> + del shunt
6. //A1 --> - del shunt
7. //A2 --> pin IAC del inversor
8. //A3 --> pin VAC inversor
9. //A4 --> pin VAC de red
10.      //A5 --> pin IAC de red
11.
12.      //CONEXIONES ADS1115
13.      //VDD --> 5V
14.      //GND --> GND
15.      //SCL --> SCL PLACA
16.      //SDA --> SDA PLACA
17.      //ADDR --> GND (Dirección por defecto)
18.      //ALRT --> SIN CONECTAR
19.      //A0 --> - DEL DIVISOR DE TENSIÓN
20.      //A1 --> + DEL DIVISOR DE TENSIÓN
21.
22.      //CONEXIONES PINES DIGITALES
23.      //2 --> Vout DEL SENSOR DE Tª
24.      //10 --> SEÑAL DE SALIDA DEL RELÉ
25.
26.      //LIBRERÍAS NECESARIAS PARA EL ADS1115 Y LA COMUNICACIÓN
   I20
27.      #include <Adafruit_BusIO_Register.h>
28.      #include <Adafruit_I2CDevice.h>
29.      #include <Adafruit_I2CRegister.h>
30.      #include <Adafruit_SPIDevice.h>
31.
32.      #include <Adafruit_ADS1X15.h>
33.
34.      #include <Wire.h>
35.
36.      //Se define la instancia para un ADC
37.      Adafruit_ADS1115 adc;
38.      const float ganancia = 0.1875F; //Depende de la ganancia
   escogida para cada ADC (GAIN_TWOTHIRDS, 6.144V de ref, 0.1875 mV
   por bit)
39.
40.      //LIBRERÍAS NECESARIAS PARA EL ENVIO DE DATOS CON ETHERNET
   AL SERVIDOR
41.      #include <SPI.h>
42.      #include <Ethernet.h>
43.
44.      EthernetClient client;
45.
46.      byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
47.      char serverName[] = "www.mienergiafotovoltaica.com";     //N
   ombre del dominio propio
48.      int serverPort = 80;     //Puerto para la transmisión de
   datos
49.      char pageAdd[64];
50.      int totalCount = 0;
51.      char Rele;
52.
53.      int FrecuenciaEnvio= 10000; //5 min = 300000 ms
```

```
54.
55.       //GANANCIA DEL DIVISOR DE TENSIÓN PARA EL VOLTAJE DC (Medir
   valores reales de las resistencias)
56.       const float R1= 14740;
57.       const float R2= 3012;
58.       const float G= R2/(R1+R2);
59.
60.
61.       //LIBRERÍA PARA CALCULAR LA CORRIENTE RMS
62.       #include <EmonLib.h>
63.
64.       EnergyMonitor emon1;     //Se crea una instancia para la
   librería --> Sensor de corriente de salida del inversor
65.       EnergyMonitor emon2;     //Se crea una instancia para la
   librería --> Sensor de corriente de red
66.
67.       //////////CALIBRACIÓN DE LOS SENSORES DE VOLTAJE AC
68.       //Los valores de vol_multi, adc_max y adc_min se obtienen
   de la calibración del sensor de voltaje AC
69.       //Calibración del sensor de voltaje de salida del inversor
70.       int adc_max=700;         //Reemplazar por valor adc_max
   entregado por el sketch: volt_ac_cal
71.       int adc_min=322;         //Reemplazar por valor adc_min
   entregado por el sketch: volt_ac_cal
72.       float volt_multi=230.1;   //Reemplazar por el "voltaje ac
   rms" entregado por un multimetro
73.       float volt_multi_p;
74.       float volt_multi_n;
75.
76.       //Calibración del sensor de voltaje de salida de la
   acometida
77.       int adc_max_red=700;         //Reemplazar por valor adc_max
   entregado por el sketch: volt_ac_cal
78.       int adc_min_red=322;         //Reemplazar por valor adc_min
   entregado por el sketch: volt_ac_cal
79.       float volt_multi_red=230.1;   //Reemplazar por el "voltaje
   ac rms" entregado por un multimetro
80.       float volt_multi_p_red;
81.       float volt_multi_n_red;
82.
83.       //VARIABLES PARA EL CÁLCULO DE POTENCIA Y ENERGÍA
84.       float Pgen, Egen;
85.       float Pimp, Pex, Pcons, Econs, Eex, Eimp;
86.       float Pautoc, Eautoc;
87.       float GENERACION, CONSUMO,
   AUTOCONSUMO;              //Almacenan los valores totales
   históricos para calcular el porcentaje de autoconsumo y la
   cobertura de la demanda
88.       float porcentaje_autoconsumo, porcentaje_excedente,
   cobertura_demanda, porcentaje_importado;
89.       float porcentaje_bateria;
90.
91.       //SE DEFINEN LAS VARIABLES DEL TEMPORIZADOR
92.       long tiempo = 0;
93.       long tiempototal, tiempoproceso;
94.
95.       extern volatile unsigned long timer0_millis;
96.       unsigned long nuevo_valor = 0;
97.       int N;
98.
99.       //CAPACIDAD DE LAS BATERÍAS EN Wh
```

```
100.      float cargamax = 2000;
101.
102.      //PIN DE CONEXIÓN PARA EL RELÉ DE GESTIÓN DE EXCEDENTES
103.      int rele = 12;
104.
105.      //PINES DE LOS LED
106.      int carga20 = 22;
107.      int carga40 = 23;
108.      int carga60 = 24;
109.      int carga80 = 25;
110.      int carga100 = 26;
111.      int cargando = 27;
112.      int descargando = 28;
113.      int sobretension = 29;
114.      int pingen = 30;
115.      int pincons = 31;
116.      int pinex = 32;
117.      int pinimp = 33;
118.
119.      void setup() {
120.
121.        Serial.begin(115200);
122.
123.      //Se desabilita la comunicación SPI con la SD del módulo
    ethernet
124.        pinMode(4,OUTPUT);
125.        digitalWrite(4,HIGH);
126.
127.        Serial.print(F("Starting ethernet..."));
128.        if(!Ethernet.begin(mac)) Serial.println(F("failed"));
129.        else Serial.println(Ethernet.localIP());
130.
131.        delay(2000);
132.        Serial.println(F("Ready"));
133.
134.      //CORRIENTE AC
135.      //IMPORTANTE: Calibrar correctamente la pinza amperimétrica
    (35 para la pinza de 30A)
136.        emon1.current(A2, 35);                    // Current:
    input pin, calibration.
137.        emon2.current(A5, 35);                    // Current:
    input pin, calibration.
138.
139.      //VOLTAJE AC
140.        volt_multi_p = volt_multi * 1.4142;   //Voltaje pico=
    Voltaje RMS * 1.4142 (Corriente Monofasica)
141.        volt_multi_n = volt_multi_p * -1;
142.
143.        volt_multi_p_red = volt_multi_red * 1.4142;   //Voltaje
    pico= Voltaje RMS * 1.4142 (Corriente Monofasica)
144.        volt_multi_n_red = volt_multi_p_red * -1;
145.
146.      //VOLTAJE DC
147.      //Configuración del adc externo ADS1115
148.        adc.begin();                      // Inicializar el adc4
    con la dirección por defecto --> pin addr conectado a GND
149.        adc.setGain(GAIN_TWOTHIRDS);        // +/- 6.144V de
    referencia --> 1 bit = 0.1875mV
150.
151.      //INICIO DE LOS CONTADORES PARA CALCULO DE ENERGÍA
```

```
152.       tiempototal= 0;                        // Tiempo cuando se
   empieza a ejecutar cada loop
153.       tiempo = 0;                            // Tiempo desde que
   se empezó a ejecutar
154.       N = 0;
155.
156.     //INICIO DE ACUMULACIÓN DE DATOS HISTÓRICOS
157.     //IMPORTANTE: MODIFICAR ESTOS VALORES SI SE RESETEA CÓDIGO
   Y NO SE QUIERE REINICIAR LA CUENTA
158.
159.       GENERACION = 0;
160.       CONSUMO = 0;
161.       AUTOCONSUMO = 0;
162.
163.       pinMode(rele, OUTPUT);
164.     }
165.
166.
167.
168.     void loop() {
169.
170.     //DECLARACIÓN DE LOS CONTADORES
171.      tiempototal = tiempo;
172.      tiempo = millis();
173.
174.
175.     //////////MEDIDAS AC: SALIDA DEL INVERSOR Y DE LA
   ACOMETIDA//////////
176.     //VOLTAJE AC
177.
178.       float Vinv, Vred =get_voltage(); //Voltage eficaz (V-RMS)
179.
180.       Serial.print("Vinv: ");
181.       Serial.print(Vinv,3);
182.       Serial.println(" VAC");
183.
184.       Serial.print("Vred: ");
185.       Serial.print(Vred,3);
186.       Serial.println(" VAC");
187.
188.
189.     //CORRIENTE AC
190.       double Iinv= emon1.calcIrms(1480);       // Cálculo de
   corriente rms I inversor
191.       double Ired = emon2.calcIrms(1480);       // Cálculo de
   corriente rms I red
192.
193.       Serial.print("Iinv= ");
194.       Serial.print(Iinv);
195.       Serial.println(" A");
196.       Serial.print("Ired= ");
197.       Serial.print(Ired);
198.       Serial.println(" A");
199.
200.     //////////MEDIDAS DC: BATERÍAS//////////
201.     //VOLTAJE DC
202.     int16_t lectura;
203.     float voltaje;
204.     float V, V_bat;
205.     int contador= 0;
206.     V_bat= 0;
```

```
207.
208.       for (int i = 0; i < 100; i ++) //toma 100 medidas por
   cada segundo y luego se calcula la media
209.       {
210.         lectura = adc.readADC_Differential_0_1();  //lectura
   del adc en modo diferencial entre A1 y A0
211.         voltaje = lectura*ganancia/1000;  //Valor de la lectura
   en voltios (escala 0-5V)
212.         V = voltaje/G;  //Voltaje real de la batería (5V -->
   30V)
213.         V_bat = V_bat + V;
214.         contador = contador +1;
215.         delay(10);
216.       }
217.
218.       V_bat = V_bat/contador;
219.       Serial.print("Vdc= ");
220.       Serial.print(V_bat);
221.       Serial.println(" V");
222.
223.
224.     //CORRIENTE DC
225.       analogReference(INTERNAL1V1);   //Referencia interna del
   arduino Mega de 1,1V
226.       delay(10);                      //Para estabilizar la
   placa después del cambio de referencia
227.
228.     //Lectura de la corriente que circula por el shunt
229.       int valor;
230.       float corriente;
231.       float shunt;
232.       float Idc=0;
233.       int cont = 0;
234.
235.       for (int i = 0; i < 100; i ++) //toma 100 medidas por
   cada segundo y luego se calcula la media
236.     {
237.       valor= analogRead(A0)-analogRead(A1);    //Diferencia
   entre los dos bornes del shunt en bits
238.       shunt = (valor*1.1/1024)*1000;          //1,1V son 1024
   bits
239.       corriente = shunt*100/75;              //100 A son 75mV
240.       Idc = Idc + corriente;
241.       cont = cont + 1;
242.       delay(10);
243.     }
244.       Idc = Idc/cont;
245.
246.       Serial.print("Idc= ");
247.       Serial.print(Idc);
248.       Serial.println(" A");
249.       analogReference(DEFAULT);             //Se reestablece al
   referencia de 5V para el adc interno
250.
251.
252.     ////CALCULOS
253.
254.       tiempoproceso= tiempo-tiempototal;
255.       float T= tiempoproceso/1000;            //Tiempo del loop
   en segundos
256.
```

```
257.       Serial.print("Tiempo proceso: ");
258.       Serial.print(tiempoproceso);
259.       Serial.println(" ms");
260.
261.
262.     //Potencia de las baterías
263.     float Pbat, Ebat;
264.
265.       Pbat = V_bat * Idc;                    //>0 si es de
   carga
266.
267.       if (Pbat > 0){
268.         digitalWrite(cargando, HIGH);
269.         digitalWrite(descargando, LOW);
270.       }else if (Pbat<0){
271.         digitalWrite(cargando, LOW);
272.         digitalWrite(descargando, HIGH);
273.       } else {
274.         //si es 0
275.         digitalWrite(cargando, LOW);
276.         digitalWrite(descargando, LOW);
277.       }
278.     //Potencia de salida del inversor
279.     float Pinv, Einv;
280.
281.       Pinv = Vinv*Iinv;                     //Se supone FP=1
282.
283.     //Potencia intercambiada con la red
284.     float Pred, Ered;
285.
286.       Pred = Vred*Ired;                     //Se supone FP=1
287.
288.     //GENERACIÓN
289.     //Cuando la potencia en las baterías es positiva (carga) o
   nula es potencia generada --> Pgen = Pinv + Pbat
290.     //Cuando la potencia en las baterías en negativa (descarga)
   no es potencia generada --> Pgen = Pinv - Pbat
291.     //Como en ese caso la Pbat es <0 --> P gen = Pinv + Pbat
292.
293.      Pgen= Pinv + Pbat;
294.
295.      Egen = Egen + Pgen*T/3600;
296.
297.     //IMPORTACIÓN, EXCEDENTES y CONSUMO
298.     //Si la potencia de la red es positiva, se está exportando
   energía (por criterio de signos)
299.     //Si l apotencia de la red es negativa, se está importando
   energía
300.       if (Pred >= 0){
301.         Pex = Pred;
302.         Pimp = 0;
303.       }
304.       else {
305.         Pimp = -Pred;     //Potencia importada en valor
   absoluto
306.         Pex = 0;
307.       }
308.
309.       Eex = Eex + Pex*T/3600;
310.       Eimp = Eimp + Pimp*T/3600;
311.
```

```
312.     //CONSUMO DE ENERGÍA
313.
314.       Pcons = Pgen - Pex + Pimp;
315.       Econs = Econs + Pcons*T/3600;
316.
317.     //AUTOCONSUMO
318.     //Cuando hay excedentes, la energía autoconsumida es la
   resta entre la generación y los excedentes
319.     //Cuando se importa energía la energía autoconsumida es
   toda la generada
320.
321.       if (Pex >0){
322.         Pautoc = Pgen - Pex;
323.       }
324.       else {
325.         Pautoc = Pgen;
326.       }
327.       Eautoc = Eautoc + Pautoc*T/3600;
328.
329.     //PORCENTAJE DE AUTOCONSUMO ACUMULADO
330.     //Generación total y consumo total acumulados
331.       GENERACION= GENERACION + Egen;
332.       CONSUMO = CONSUMO + Econs;
333.       AUTOCONSUMO = AUTOCONSUMO + Eautoc;
334.
335.       porcentaje_autoconsumo = (AUTOCONSUMO/GENERACION)*100;
336.       porcentaje_excedente = 100 - porcentaje_autoconsumo;
337.
338.       cobertura_demanda = (AUTOCONSUMO/CONSUMO)*100;
339.       porcentaje_importado = 100 - cobertura_demanda;
340.
341.     //ESTADO DE CARGA DE LAS BATERÍAS
342.     //Se calcula el nivel de carga de las baterías en función
   de la energía almacenada
343.
344.       Ebat = Ebat + Pbat*T/3600;
345.
346.       if (Ebat > cargamax) {
347.         Ebat = cargamax;}
348.
349.       porcentaje_bateria = (Ebat/cargamax)*100;
350.
351.       if (porcentaje_bateria > 80){
352.         digitalWrite(carga20, HIGH);
353.         digitalWrite(carga40, HIGH);
354.         digitalWrite(carga60, HIGH);
355.         digitalWrite(carga80, HIGH);
356.         digitalWrite(carga100, HIGH);
357.         }
358.       else if (80 >= porcentaje_bateria > 60){
359.         digitalWrite(carga20, HIGH);
360.         digitalWrite(carga40, HIGH);
361.         digitalWrite(carga60, HIGH);
362.         digitalWrite(carga80, HIGH);
363.         digitalWrite(carga100, LOW);
364.       } else if (60 >= porcentaje_bateria > 40){
365.         digitalWrite(carga20, HIGH);
366.         digitalWrite(carga40, HIGH);
367.         digitalWrite(carga60, HIGH);
368.         digitalWrite(carga80, LOW);
369.         digitalWrite(carga100, LOW);
```

```
370.       }
371.
372.         else if (40 >= porcentaje_bateria > 20){
373.         digitalWrite(carga20, HIGH);
374.         digitalWrite(carga40, HIGH);
375.         digitalWrite(carga60, LOW);
376.         digitalWrite(carga80, LOW);
377.         digitalWrite(carga100, LOW);
378.       }
379.         else if (20 >= porcentaje_bateria ){
380.         digitalWrite(carga20, HIGH);
381.         digitalWrite(carga40, LOW);
382.         digitalWrite(carga60, LOW);
383.         digitalWrite(carga80, LOW);
384.         digitalWrite(carga100, LOW);
385.       }
386.       else if (porcentaje_bateria = 0){
387.         digitalWrite(carga20, LOW);
388.         digitalWrite(carga40, LOW);
389.         digitalWrite(carga60, LOW);
390.         digitalWrite(carga80, LOW);
391.         digitalWrite(carga100, LOW);
392.       }
393.
394.     //GESTIÓN DE EXCEDENTES
395.       if (Pex >0){
396.         digitalWrite(rele, HIGH); //Cuando hay excedentes se
   activa la salida del relé
397.       }
398.       else {
399.         digitalWrite(rele, LOW);
400.       }
401.     Serial.print("Estado del relé: ");
402.     Serial.println(Rele);
403.
404.      if (tiempototal > FrecuenciaEnvio){
405.       //ENVÍO DE LOS DATOS ALMACENADOS EN LOS ÚLTIMOS 5 MIN
406.       //Se envía Egen, Econs, Eimp, Eex, Eautoc,
   porcentaje_autoconsumo, porcentaje_excedentes, cobertura_demanda
407.
408.         sprintf(pageAdd,"/databaseconnect.php?lectura=%s",total
   Count);
409.
410.       //SI FALLA LA CONEXIÓN AVISA, SI NO SE LLAMA getPage
411.         if(!getPage(serverName,serverPort,pageAdd)) Serial.prin
   t(F("Fail "));
412.         else Serial.print(F("Pass "));
413.         totalCount++;
414.         Serial.println(totalCount,DEC);
415.
416.
417.       //RESETEO DE LOS VALORES DE ENERGÍA ACUMULADOS
418.       //Valores de energía calculados en el último periodo de
   toma de datos
419.       //Los valores totales no se resetean porque sirven para
   acumular datos históricos
420.       Egen =0;
421.       Eimp =0;
422.       Eex =0;
423.       Econs =0;
424.       Eautoc =0;
```

```
425.
426.
427.        //RESETEO DEL CONTADOR
428.        setMillis(nuevo_valor);
429.        tiempo = millis();
430.        tiempototal = tiempo;
431.       }
432.
433.     }
434.
435.
436.
437.
438.
439.     //CÁLCULO DEL VOLTAJE AC RMS
440.     //IMPORTANTE: NECESITA 5V COMO REFERENCIA ANALÓGICA
441.     //DEVUELVE EL VOLTAJE DE SALIDA DEL INVERSOR Y EL VOLTAJE
   DE SALIDA DE LA ACOMETIDA
442.     float get_voltage(void)
443.     {
444.       //analogReference(DEFAULT); //POR SI SE HA CAMBIADO EN
   OTRO MOMENTO
445.       float adc_inv, adc_red;
446.       float volt_inst=0;
447.       float volt_inst_red=0;
448.       float Sumatoria=0;
449.       float Sum=0;
450.       float V_inv, V_red;
451.       long tiempo_init=millis();
452.       int N=0;
453.
454.       while( (millis() - tiempo_init) < 600)//Duración 0.6
   segundos(Aprox. 30 ciclos de 50Hz)
455.       {
456.
457.         adc_inv = analogRead(A3);    //Entrada sensor de
   voltaje del inversor
458.         adc_red = analogRead(A4);    //Entrada sensor de
   voltaje de la red
459.
460.         volt_inst = map(adc_inv,adc_min,adc_max,volt_multi_n,vo
   lt_multi_p);
461.         volt_inst_red = map(adc_red,adc_min_red,adc_max_red,vol
   t_multi_n_red,volt_multi_p_red);
462.
463.         Sumatoria = Sumatoria+sq(volt_inst);    //Sumatoria de
   Cuadrados
464.         Sum = Sum+sq(volt_inst_red);            //Sumatoria de
   Cuadrados
465.
466.         N = N+1;
467.         delay(1);
468.       }
469.
470.       V_inv=sqrt((Sumatoria)/N);  //ecuación del RMS
471.       V_red=sqrt((Sum)/N);        //ecuación del RMS
472.
473.       return V_inv, V_red;
474.     }
475.
476.     //RESETEO DEL CONTADOR para que no se desborde
```

```
477.     void setMillis(unsigned long nuevo_valor){
478.       uint8_t oldSREG = SREG;
479.       cli();
480.       timer0_millis = nuevo_valor;
481.       SREG = oldSREG;
482.     }
483.
484.
485.     //ENVÍO DE LOS DATOS
486.     byte getPage (char* domainBuffer,int thisPort,char* page)
487.     {
488.       int inChar;
489.       char outBuf[64];
490.
491.       Serial.print(F("connecting..."));
492.
493.
494.       if(client.connect(domainBuffer,thisPort) == 1)
495.       {
496.         Serial.print(F("connected to "));
497.         Serial.println(domainBuffer);
498.         Serial.println();
499.
500.         Serial.println(Egen);
501.         Serial.println(Econs);
502.         Serial.println(Eautoc);
503.         Serial.println(Eex);
504.         Serial.println(Eimp);
505.
506.         client.print("GET /databaseconnect.php?generacion=");
507.         client.print(Egen);
508.         client.print("&consumo=");
509.         client.print(Econs);
510.         client.print("&autoconsumo=");
511.         client.print(Eautoc);
512.         client.print("&exportado=");
513.         client.print(Eex);
514.         client.print("&importado=");
515.         client.print(Eimp);
516.         client.print("&porcentaje_autoconsumo=");
517.         client.print(porcentaje_autoconsumo);
518.         client.print("&porcentaje_excedente=");
519.         client.print(porcentaje_excedente);
520.         client.print("&cobertura_demanda=");
521.         client.print(cobertura_demanda);
522.         client.print("&porcentaje_importado=");
523.         client.print(porcentaje_importado);
524.         client.print("&porcentaje_bateria=");
525.         client.print(porcentaje_bateria);
526.         client.println(" HTTP/1.1");
527.
528.         sprintf(outBuf,"Host: %s",serverName);
529.         client.println(outBuf);
530.         client.println(F("Connection: close\r\n"));
531.
532.       }
533.       else
534.       {
535.         Serial.println(F("failed"));
536.         return 0;
537.       }
```

```
538.
539.        int connectLoop = 0;
540.
541.        while(client.connected())
542.        {
543.          while(client.available())
544.          {
545.            inChar = client.read();
546.            Serial.write(inChar);
547.            connectLoop = 0;
548.          }
549.
550.          delay(1);
551.          connectLoop++;
552.          if(connectLoop > 10000)
553.          {
554.            Serial.println();
555.            Serial.println(F("Timeout"));
556.            client.stop();
557.          }
558.        }
559.
560.        Serial.println();
561.        Serial.println(F("disconnecting."));
562.        client.stop();
563.        return 1;
564.      }
```

ANEXO III: Programación para autoconsumo trifásico sin baterías

```
1. ////////// AUTOCONSUMO TRIFÁSICO //////////
2. ////////////////////////////////////////////////////////
3.
4. //CONEXIONES ADC INTERNO ARDUINO
5. //A2 --> pin IAC del inversor (FASE R)
6. //A3 --> pin IAC del inversor (FASE S)
7. //A4 --> pin IAC del inversor (FASE T)
8. //A5 --> pin IAC de red (FASE R)
9. //A6 --> pin IAC de red (FASE S)
10.      //A7 --> pin IAC de red (FASE T)
11.      //A8 --> pin VAC inversor (FASE R)
12.      //A9 --> pin VAC inversor (FASE S)
13.      //A10 --> pin VAC inversor (FASE T)
14.      //A11 --> pin VAC de red (FASE R)
15.      //A12 --> pin VAC de red (FASE S)
16.      //A13 --> pin VAC de red (FASE T)
17.
18.      //CONEXIONES PINES DIGITALES
19.      //2 --> Vout DEL SENSOR DE Tª
20.      //50 --> SEÑAL DE SALIDA DEL RELÉ (FASE R)
21.      //51 --> SEÑAL DE SALIDA DEL RELÉ (FASE S)
22.      //52 --> SEÑAL DE SALIDA DEL RELÉ (FASE T)
23.
24.      //LIBRERÍAS NECESARIAS PARA EL ENVIO DE DATOS CON ETHERNET
   AL SERVIDOR
25.      #include <SPI.h>
26.      #include <Ethernet.h>
27.
28.      EthernetClient client;
29.
30.      byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
31.      char serverName[] = "www.mienergiafotovoltaica.com";     //N
   ombre del dominio propio
32.      int serverPort = 80;     //Puerto para la transmisión de
   datos
33.      char pageAdd[64];
34.      int totalCount = 0;
35.      int FrecuenciaEnvio= 300000; //5 min = 300000 ms
36.
37.
38.      //LIBRERÍA PARA CALCULAR LA CORRIENTE RMS
39.      #include <EmonLib.h>
40.      //Se crean instancias para la librería (una para cada
   sensor de corriente)
41.      EnergyMonitor inv1;     //Se crea una instancia para la
   librería --> Sensor de corriente de salida del inversor
42.      EnergyMonitor inv2;     //Se crea una instancia para la
   librería --> Sensor de corriente de salida del inversor
43.      EnergyMonitor inv3;     //Se crea una instancia para la
   librería --> Sensor de corriente de salida del inversor
44.      EnergyMonitor red1;     //Se crea una instancia para la
   librería --> Sensor de corriente de red
45.      EnergyMonitor red2;     //Se crea una instancia para la
   librería --> Sensor de corriente de red
46.      EnergyMonitor red3;     //Se crea una instancia para la
   librería --> Sensor de corriente de red
47.
48.      //////////CALIBRACIÓN DE LOS SENSORES DE VOLTAJE AC
```

```
49.      //Los valores de vol_multi, adc_max y adc_min se obtienen
   de la calibración del sensor de voltaje AC
50.      //Reemplazar los valores adc_max, adc_min y volt_multi por
   los entregados por el sketch: volt_ac_cal para cada sensor
51.
52.      //Sensores de voltaje fase-neutro a la salida del inversor
53.      int adc_max_inv1=700;
54.      int adc_min_inv1=322;
55.      float volt_multi_inv1=230.1;
56.      float volt_multi_p_inv1;
57.      float volt_multi_n_inv1;
58.
59.      int adc_max_inv2=700;
60.      int adc_min_inv2=322;
61.      float volt_multi_inv2=230.1;
62.      float volt_multi_p_inv2;
63.      float volt_multi_n_inv2;
64.
65.      int adc_max_inv3=700;
66.      int adc_min_inv3=322;
67.      float volt_multi_inv3=230.1;
68.      float volt_multi_p_inv3;
69.      float volt_multi_n_inv3;
70.
71.      //Sensores de voltaje fase-neutro a la salida de la
   acometida
72.      int adc_max_red1=700;
73.      int adc_min_red1=322;
74.      float volt_multi_red1=230.1;
75.      float volt_multi_p_red1;
76.      float volt_multi_n_red1;
77.
78.      int adc_max_red2=700;
79.      int adc_min_red2=322;
80.      float volt_multi_red2=230.1;
81.      float volt_multi_p_red2;
82.      float volt_multi_n_red2;
83.
84.      int adc_max_red3=700;
85.      int adc_min_red3=322;
86.      float volt_multi_red3=230.1;
87.      float volt_multi_p_red3;
88.      float volt_multi_n_red3;
89.
90.      //VARIABLES PARA EL CÁLCULO DE POTENCIA Y ENERGÍA
91.      float Pgen, Egen;
92.      float Pimp, Pex, Pcons, Econs, Eex, Eimp;
93.      float Pautoc, Eautoc;
94.      float GENERACION, CONSUMO,
   AUTOCONSUMO;             //Almacenan los valores totales
   históricos para calcular el porcentaje de autoconsumo y la
   cobertura de la demanda
95.      float porcentaje_autoconsumo, porcentaje_excedente,
   cobertura_demanda, porcentaje_importado;
96.      float porcentaje_bateria;
97.      float Ebat;
98.
99.      //SE DEFINEN LAS VARIABLES DEL TEMPORIZADOR
100.     long tiempo = 0;
101.     long tiempototal, tiempoproceso;
102.
```

```
103.     extern volatile unsigned long timer0_millis;
104.     unsigned long nuevo_valor = 0;
105.     int N;
106.
107.     //PINES DE CONEXIÓN PARA LOS RELÉS DE GESTIÓN DE EXCEDENTES
108.     int rele_r = 50;
109.     int rele_s = 51;
110.     int rele_t = 52;
111.
112.
113.     void setup() {
114.
115.       Serial.begin(115200);
116.
117.     //Se desabilita la comunicación SPI con la SD del módulo
   ethernet
118.       pinMode(4,OUTPUT);
119.       digitalWrite(4,HIGH);
120.
121.       Serial.print(F("Starting ethernet..."));
122.       if(!Ethernet.begin(mac)) Serial.println(F("failed"));
123.       else Serial.println(Ethernet.localIP());
124.
125.       delay(2000);
126.       Serial.println(F("Ready"));
127.
128.     //CORRIENTE AC
129.     //IMPORTANTE: Calibrar correctamente la pinza amperimétrica
   (35 para la pinza de 30A)
130.       inv1.current(A2, 35);                    // Current:
   input pin, calibration.
131.       inv2.current(A3, 35);                    // Current:
   input pin, calibration.
132.       inv3.current(A4, 35);                    // Current:
   input pin, calibration.
133.       red1.current(A5, 35);                    // Current:
   input pin, calibration.
134.       red2.current(A6, 35);                    // Current:
   input pin, calibration.
135.       red3.current(A7, 35);                    // Current:
   input pin, calibration.
136.
137.
138.     //VOLTAJE AC
139.       volt_multi_p_inv1 = volt_multi_inv1 * 1.4142;   //Voltaje
   pico= Voltaje RMS * 1.4142
140.       volt_multi_n_inv1 = volt_multi_p_inv1 * -1;
141.
142.       volt_multi_p_inv2 = volt_multi_inv2 * 1.4142;   //Voltaje
   pico= Voltaje RMS * 1.4142
143.       volt_multi_n_inv2 = volt_multi_p_inv2 * -1;
144.
145.       volt_multi_p_inv3 = volt_multi_inv3 * 1.4142;   //Voltaje
   pico= Voltaje RMS * 1.4142
146.       volt_multi_n_inv3 = volt_multi_p_inv3 * -1;
147.
148.       volt_multi_p_red1 = volt_multi_red1 * 1.4142;   //Voltaje
   pico= Voltaje RMS * 1.4142
149.       volt_multi_n_red1 = volt_multi_p_red1 * -1;
150.
```

```
151.       volt_multi_p_red2 = volt_multi_red2 * 1.4142;   //Voltaje
   pico= Voltaje RMS * 1.4142
152.       volt_multi_n_red2 = volt_multi_p_red2 * -1;
153.
154.       volt_multi_p_red3 = volt_multi_red3 * 1.4142;   //Voltaje
   pico= Voltaje RMS * 1.4142
155.       volt_multi_n_red3 = volt_multi_p_red3 * -1;
156.
157.     //INICIO DE LOS CONTADORES PARA CALCULO DE POTENCIAS
158.       tiempototal= 0;                    // Tiempo cuando se
   empieza a ejecutar cada loop
159.       tiempo = 0;                        // Tiempo desde que
   se empezó a ejecutar
160.       N = 0;
161.
162.     //INICIO DE ACUMULACIÓN DE DATOS HISTÓRICOS
163.     //IMPORTANTE: MODIFICAR ESTOS VALORES SI SE RESETEA CÓDIGO
   Y NO SE QUIERE REINICIAR LA CUENTA
164.
165.       GENERACION = 0;
166.       CONSUMO = 0;
167.       AUTOCONSUMO = 0;
168.
169.       pinMode(rele_r, OUTPUT);
170.       pinMode(rele_s, OUTPUT);
171.       pinMode(rele_t, OUTPUT);
172.
173.     }
174.
175.
176.
177.     void loop() {
178.
179.     //DECLARACIÓN DE LOS CONTADORES
180.      tiempototal = tiempo;
181.      tiempo = millis();
182.
183.
184.     //////////MEDIDAS AC: SALIDA DEL INVERSOR Y DE LA
   ACOMETIDA//////////
185.     //VOLTAJE AC
186.
187.       float Vinv1, Vinv2, Vinv3, Vred1, Vred2,
   Vred3 =get_voltage(); //Voltage eficaz (V-RMS)
188.
189.     //DESEQUILIBRIO DE FASES
190.     //Desequilibrio a la salida del inversor
191.     float Vmedia = (Vinv1 + Vinv2 + Vinv3)/3;
192.     float desequilibrio;
193.
194.       desequilibrio = (max(max(abs(Vinv1-Vinv2),abs(Vinv1-
   Vinv3)),abs(Vinv2-Vinv3))/Vmedia)*100;
195.
196.     //Desequilibrio a la salida del inversor
197.     float Vmedia_red = (Vred1 + Vred2 + Vred3)/3;
198.     float desequilibrio;
199.
200.       desequilibrio_red = (max(max(abs(Vred1-Vred2),abs(Vred1-
   Vred3)),abs(Vred2-Vred3))/Vmedia_red)*100;
201.
202.       if (desequilibrio > 2){
```

```
203.          digitalWrite(11, HIGH); //Enciende el led de alarma
204.        }else{
205.          digitalWrite(11, LOW);
206.        }
207.        if (desequilibrio > 2){
208.          digitalWrite(12, HIGH); //Enciende el led de alarma
209.        }else{
210.          digitalWrite(12, LOW);
211.        }
212.
213.        Serial.println("VOLTAJES DE FASE A LA SALIDA DEL
    INVERSOR");
214.        Serial.print("Fase R: ");
215.        Serial.print(Vinv1,3);
216.        Serial.println(" V");
217.        Serial.print("Fase S: ");
218.        Serial.print(Vinv2,3);
219.        Serial.println(" V");
220.        Serial.print("Fase T: ");
221.        Serial.print(Vinv3,3);
222.        Serial.println(" V");
223.
224.        Serial.println("VOLTAJES DE FASE A LA SALIDA DE LA
    ACOMETIDA");
225.        Serial.print("Fase R: ");
226.        Serial.print(Vred1,3);
227.        Serial.println(" V");
228.        Serial.print("Fase s: ");
229.        Serial.print(Vred2,3);
230.        Serial.println(" V");
231.        Serial.print("Fase t: ");
232.        Serial.print(Vred3,3);
233.        Serial.println(" V");
234.
235.      //CORRIENTE AC
236.      //Corrientes a la salida del inversor
237.        double Iinv1= inv1.calcIrms(1480);
238.        double Iinv2= inv2.calcIrms(1480);
239.        double Iinv3= inv3.calcIrms(1480);
240.
241.      //Corrientes a la salida de la acometida
242.        double Ired1 = red1.calcIrms(1480);
243.        double Ired2 = red2.calcIrms(1480);
244.        double Ired3 = red3.calcIrms(1480);
245.
246.        Serial.println("CORRIENTES DE FASE A LA SALIDA DEL
    INVERSOR");
247.        Serial.print("Fase R: ");
248.        Serial.print(Iinv1,3);
249.        Serial.println(" A");
250.        Serial.print("Fase S: ");
251.        Serial.print(Iinv2,3);
252.        Serial.println(" A");
253.        Serial.print("Fase T: ");
254.        Serial.print(Iinv3,3);
255.        Serial.println(" A");
256.
257.        Serial.println("CORRIENTES DE FASE A LA SALIDA DE LA
    ACOMETIDA");
258.        Serial.print("Fase R: ");
259.        Serial.print(Ired1,3);
```

```
260.       Serial.println(" A");
261.       Serial.print("Fase S: ");
262.       Serial.print(Ired2,3);
263.       Serial.println(" A");
264.       Serial.print("Fase T: ");
265.       Serial.print(Ired3,3);
266.       Serial.println(" A");
267.
268.     ////////////////////////////////////////////////////////////
   ////////////////////////////
269.     ////CÁLCULOS
270.       tiempoproceso= tiempo-tiempototal;    //Se calcula el
   tiempo que ha pasado desde la última vez que se ejecutó el loop
271.       Serial.print("Tiempo proceso: ");
272.       Serial.print(tiempoproceso);
273.       Serial.println(" ms");
274.
275.     float T= tiempoproceso/1000;            //Tiempo del loop
   en segundos
276.
277.     //Potencia de salida del inversor
278.     float Pinv, Einv;
279.
280.       Pinv = Vinv1*Iinv1 + Vinv2*Iinv2 + Vinv3*Iinv3;
            //Se supone FP=1
281.
282.     //Potencia intercambiada con la red
283.     float Pred, Ered;
284.
285.       Pred = Vred1*Ired1 + Vred2*Ired2 + Vred3*Ired3;
            //Se supone FP=1
286.
287.     //GENERACIÓN
288.     //Toda la energía que se genera atraviesa el inversor
   (quitando las pérdidas)
289.      Pgen= Pinv;
290.
291.      Egen = Egen + Pgen*T/3600;
292.
293.     //IMPORTACIÓN, EXCEDENTES y CONSUMO
294.     //Si la potencia de la red es positiva, se está exportando
   energía (por criterio de signos)
295.     //Si se está importando energía y a la vez generando, el
   consumo total es la suma de ambos conceptos
296.
297.       if (Pred >= 0){
298.         Pex = Pred;
299.         Pimp = 0;
300.       }
301.       else {
302.         Pimp = -Pred;     //Potencia importada en valor
   absoluto
303.         Pex = 0;
304.       }
305.       Pcons = Pgen - Pex + Pimp;
306.       Eex = Eex + Pex*T/3600;
307.       Eimp = Eimp + Pimp*T/3600;
308.       Econs = Econs + Pcons*T/3600;
309.
310.     //AUTOCONSUMO
```

```
311.     //Cuando hay excedentes, la energía autoconsumida es la
   resta entre la generación y los excedentes
312.     //Cuando se importa energía la energía autoconsumida es
   toda la generada
313.
314.       if (Pex >0){
315.         Pautoc = Pgen - Pex;
316.       }
317.       else {
318.         Pautoc = Pgen;
319.       }
320.       Eautoc = Eautoc + Pautoc*T/3600;
321.
322.     //PORCENTAJE DE AUTOCONSUMO ACUMULADO
323.     //Generación total y consumo total acumulados
324.       GENERACION= GENERACION + Egen;
325.       CONSUMO = CONSUMO + Econs;
326.       AUTOCONSUMO = AUTOCONSUMO + Eautoc;
327.
328.       porcentaje_autoconsumo = (AUTOCONSUMO/GENERACION)*100;
329.       porcentaje_excedente = 100 - porcentaje_autoconsumo;
330.
331.       cobertura_demanda = (AUTOCONSUMO/CONSUMO)*100;
332.       porcentaje_importado = 100 - cobertura_demanda;
333.
334.
335.
336.      if (tiempototal > FrecuenciaEnvio){
337.       //ENVÍO DE LOS DATOS ALMACENADOS EN LOS ÚLTIMOS 5 MIN
338.       //Se envía Egen, Econs, Eimp, Eex, Eautoc,
   porcentaje_autoconsumo,
339.       //porcentaje_excedente, cobertura_demanda y
   porcentaje_importado
340.
341.         sprintf(pageAdd,"/databaseconnect.php?lectura=%s",total
   Count);
342.
343.       //SI FALLA LA CONEXIÓN AVISA, SI NO SE LLAMA getPage
344.         if(!getPage(serverName,serverPort,pageAdd)) Serial.prin
   t(F("Fail "));
345.         else Serial.print(F("Pass "));
346.         totalCount++;
347.         Serial.println(totalCount,DEC);
348.
349.       //RESETEO DE LOS VALORES DE ENERGÍA ACUMULADOS
350.       Egen =0;
351.       Eimp =0;
352.       Eex =0;
353.       Econs =0;
354.       Eautoc =0;
355.
356.      }
357.
358.     }
359.
360.
361.
362.
363.
364.     //CÁLCULO DEL VOLTAJE AC RMS
365.     //IMPORTANTE: NECESITA 5V COMO REFERENCIA ANALÓGICA
```

```
366.     //DEVUELVE EL VOLTAJE DE SALIDA DEL INVERSOR Y EL VOLTAJE
   DE SALIDA DE LA ACOMETIDA
367.     float get_voltage(void)
368.     {
369.       //analogReference(DEFAULT); //POR SI SE HA CAMBIADO EN
   OTRO MOMENTO
370.       float adc_inv1, adc_inv2, adc_inv3, adc_red1, adc_red2,
   adc_red3;
371.       float volt_inst1=0;
372.       float volt_inst2=0;
373.       float volt_inst3=0;
374.       float volt_inst_red1=0;
375.       float volt_inst_red2=0;
376.       float volt_inst_red3=0;
377.       float Sum1=0;
378.       float Sum2=0;
379.       float Sum3=0;
380.       float Sum4=0;
381.       float Sum5=0;
382.       float Sum6=0;
383.       float V_inv1, V_inv2, V_inv3, V_red1, V_red2, V_red3;
384.       long tiempo_init=millis();
385.       int N=0;
386.
387.       while( (millis() - tiempo_init) < 600)//Duración 0.6
   segundos(Aprox. 30 ciclos de 50Hz)
388.       {
389.
390.         adc_inv1 = analogRead(A8);    //Entrada sensor de
   voltaje del inversor
391.         adc_inv2 = analogRead(A9);    //Entrada sensor de
   voltaje del inversor
392.         adc_inv3 = analogRead(A10);    //Entrada sensor de
   voltaje del inversor
393.
394.         adc_red1 = analogRead(A11);    //Entrada sensor de
   voltaje de la red
395.         adc_red2 = analogRead(A12);    //Entrada sensor de
   voltaje de la red
396.         adc_red3 = analogRead(A13);    //Entrada sensor de
   voltaje de la red
397.
398.         volt_inst1 = map(adc_inv1,adc_min_inv1,adc_max_inv1,vol
   t_multi_n_inv1,volt_multi_p_inv1);
399.         volt_inst2 = map(adc_inv2,adc_min_inv2,adc_max_inv2,vol
   t_multi_n_inv2,volt_multi_p_inv2);
400.         volt_inst3 = map(adc_inv3,adc_min_inv3,adc_max_inv3,vol
   t_multi_n_inv3,volt_multi_p_inv3);
401.
402.         volt_inst_red1 = map(adc_red1,adc_min_red1,adc_max_red1
   ,volt_multi_n_red1,volt_multi_p_red1);
403.         volt_inst_red2 = map(adc_red2,adc_min_red2,adc_max_red2
   ,volt_multi_n_red2,volt_multi_p_red2);
404.         volt_inst_red3 = map(adc_red3,adc_min_red3,adc_max_red3
   ,volt_multi_n_red3,volt_multi_p_red3);
405.
406.     //Sumatoria de Cuadrados
407.         Sum1 = Sum1+sq(volt_inst1);
408.         Sum2 = Sum2+sq(volt_inst2);
409.         Sum3 = Sum3+sq(volt_inst3);
410.
```

```
411.          Sum4 = Sum4+sq(volt_inst_red1);
412.          Sum5 = Sum5+sq(volt_inst_red2);
413.          Sum6 = Sum6+sq(volt_inst_red3);
414.
415.          N = N+1;
416.          delay(1);
417.        }
418.      //ecuación del RMS
419.        V_inv1=sqrt((Sum1)/N);
420.        V_inv2=sqrt((Sum2)/N);
421.        V_inv3=sqrt((Sum3)/N);
422.        V_red1=sqrt((Sum4)/N);
423.        V_red2=sqrt((Sum5)/N);
424.        V_red3=sqrt((Sum6)/N);
425.
426.        return V_inv1, V_inv2, V_inv3, V_red1, V_red2, V_red3;
427.      }
428.
429.      //RESETEO DEL CONTADOR para que no se desborde
430.      void setMillis(unsigned long nuevo_valor){
431.        uint8_t oldSREG = SREG;
432.        cli();
433.        timer0_millis = nuevo_valor;
434.        SREG = oldSREG;
435.      }
436.
437.
438.      //ENVÍO DE LOS DATOS
439.      byte getPage (char* domainBuffer,int thisPort,char* page)
440.      { int inChar;
441.        char outBuf[64];
442.        Serial.print(F("connecting..."));
443.        if(client.connect(domainBuffer,thisPort) == 1)
444.        { Serial.print(F("connected to "));
445.          Serial.println(domainBuffer);
446.          Serial.println();
447.          client.print("GET /databaseconnect.php?generacion=");
448.          client.print(Egen);
449.          client.print("&consumo=");
450.          client.print(Econs);
451.          client.print("&autoconsumo=");
452.          client.print(Eautoc);
453.          client.print("&exportado=");
454.          client.print(Eex);
455.          client.print("&importado=");
456.          client.print(Eimp);
457.          client.print("&porcentaje_autoconsumo=");
458.          client.print(porcentaje_autoconsumo);
459.          client.print("&porcentaje_excedente=");
460.          client.print(porcentaje_excedente);
461.          client.print("&cobertura_demanda=");
462.          client.print(cobertura_demanda);
463.          client.print("&porcentaje_importado=");
464.          client.print(porcentaje_importado);
465.          client.println(" HTTP/1.1");
466.
467.          sprintf(outBuf,"Host: %s",serverName);
468.          client.println(outBuf);
469.          client.println(F("Connection: close\r\n")); }
470.        else
471.        { Serial.println(F("failed"));
```

```
472.         return 0; }
473.       int connectLoop = 0;
474.       while(client.connected())
475.       {
476.         while(client.available())
477.         {
478.           inChar = client.read();
479.           Serial.write(inChar);
480.           connectLoop = 0; }
481.         delay(1);
482.         connectLoop++;
483.         if(connectLoop > 10000)
484.         {
485.           Serial.println();
486.           Serial.println(F("Timeout"));
487.           client.stop(); } }
488.       Serial.println();
489.       Serial.println(F("disconnecting."));
490.       client.stop();
491.       return 1;
492.     }
```

ANEXO IV: Programación para autoconsumo monofásico sin baterías

```
1. ////////// AUTOCONSUMO CON BATERÍAS MONOFÁSICO //////////
2. ////////////////////////////////////////////////////////
3.
4. //CONEXIONES ADC INTERNO ARDUINO
5. //A2 --> pin IAC del inversor
6. //A3 --> pin VAC inversor
7. //A4 --> pin VAC de red
8. //A5 --> pin IAC de red
9.
10.
11.      //CONEXIONES PINES DIGITALES
12.      //10 --> SEÑAL DE SALIDA DEL RELÉ
13.
14.      //LIBRERÍAS NECESARIAS PARA EL ENVIO DE DATOS CON ETHERNET
   AL SERVIDOR
15.      #include <SPI.h>
16.      #include <Ethernet.h>
17.
18.      EthernetClient client;
19.
20.      byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
21.      char serverName[] = "www.mienergiafotovoltaica.com";     //N
   ombre del dominio propio
22.      int serverPort = 80;     //Puerto para la transmisión de
   datos
23.      char pageAdd[64];
24.      int totalCount = 0;
25.      char Rele;
26.
27.      int FrecuenciaEnvio= 10000; //5 min = 300000 ms
28.
29.      //LIBRERÍA PARA CALCULAR LA CORRIENTE RMS
30.      #include <EmonLib.h>
31.
32.      EnergyMonitor emon1;     //Se crea una instancia para la
   librería --> Sensor de corriente de salida del inversor
33.      EnergyMonitor emon2;     //Se crea una instancia para la
   librería --> Sensor de corriente de red
34.
35.      //////////CALIBRACIÓN DE LOS SENSORES DE VOLTAJE AC
36.      //Los valores de vol_multi, adc_max y adc_min se obtienen
   de la calibración del sensor de voltaje AC
37.      //Calibración del sensor de voltaje de salida del inversor
38.      int adc_max=700;         //Reemplazar por valor adc_max
   entregado por el sketch: volt_ac_cal
39.      int adc_min=322;         //Reemplazar por valor adc_min
   entregado por el sketch: volt_ac_cal
40.      float volt_multi=230.1;    //Reemplazar por el "voltaje ac
   rms" entregado por un multimetro
41.      float volt_multi_p;
42.      float volt_multi_n;
43.
44.      //Calibración del sensor de voltaje de salida de la
   acometida
45.      int adc_max_red=700;         //Reemplazar por valor adc_max
   entregado por el sketch: volt_ac_cal
46.      int adc_min_red=322;         //Reemplazar por valor adc_min
   entregado por el sketch: volt_ac_cal
```

```
47.       float volt_multi_red=230.1;   //Reemplazar por el "voltaje
   ac rms" entregado por un multimetro
48.       float volt_multi_p_red;
49.       float volt_multi_n_red;
50.
51.       //VARIABLES PARA EL CÁLCULO DE POTENCIA Y ENERGÍA
52.       float Pgen, Egen;
53.       float Pimp, Pex, Pcons, Econs, Eex, Eimp;
54.       float Pautoc, Eautoc;
55.       float GENERACION, CONSUMO,
   AUTOCONSUMO;              //Almacenan los valores totales
   históricos para calcular el porcentaje de autoconsumo y la
   cobertura de la demanda
56.       float porcentaje_autoconsumo, porcentaje_excedente,
   cobertura_demanda, porcentaje_importado;
57.       float porcentaje_bateria;
58.
59.       //SE DEFINEN LAS VARIABLES DEL TEMPORIZADOR
60.       long tiempo = 0;
61.       long tiempototal, tiempoproceso;
62.
63.       extern volatile unsigned long timer0_millis;
64.       unsigned long nuevo_valor = 0;
65.       int N;
66.
67.       //CAPACIDAD DE LAS BATERÍAS EN Wh
68.       float cargamax = 2000;
69.
70.       //PIN DE CONEXIÓN PARA EL RELÉ DE GESTIÓN DE EXCEDENTES
71.       int rele = 12;
72.
73.       //PINES DE LOS LED
74.       int carga20 = 22;
75.       int carga40 = 23;
76.       int carga60 = 24;
77.       int carga80 = 25;
78.       int carga100 = 26;
79.       int cargando = 27;
80.       int descargando = 28;
81.       int sobretension = 29;
82.       int pingen = 30;
83.       int pincons = 31;
84.       int pinex = 32;
85.       int pinimp = 33;
86.
87.       void setup() {
88.
89.         Serial.begin(115200);
90.
91.       //Se desabilita la comunicación SPI con la SD del módulo
   ethernet
92.         pinMode(4,OUTPUT);
93.         digitalWrite(4,HIGH);
94.
95.         Serial.print(F("Starting ethernet..."));
96.         if(!Ethernet.begin(mac)) Serial.println(F("failed"));
97.         else Serial.println(Ethernet.localIP());
98.
99.         delay(2000);
100.        Serial.println(F("Ready"));
101.
```

```
102.     //CORRIENTE AC
103.     //IMPORTANTE: Calibrar correctamente la pinza amperimétrica
   (35 para la pinza de 30A)
104.       emon1.current(A2, 35);                    // Current:
   input pin, calibration.
105.       emon2.current(A5, 35);                    // Current:
   input pin, calibration.
106.
107.     //VOLTAJE AC
108.       volt_multi_p = volt_multi * 1.4142;   //Voltaje pico=
   Voltaje RMS * 1.4142 (Corriente Monofasica)
109.       volt_multi_n = volt_multi_p * -1;
110.
111.       volt_multi_p_red = volt_multi_red * 1.4142;   //Voltaje
   pico= Voltaje RMS * 1.4142 (Corriente Monofasica)
112.       volt_multi_n_red = volt_multi_p_red * -1;
113.
114.     //INICIO DE LOS CONTADORES PARA CALCULO DE ENERGÍA
115.       tiempototal= 0;                     // Tiempo cuando se
   empieza a ejecutar cada loop
116.       tiempo = 0;                         // Tiempo desde que
   se empezó a ejecutar
117.       N = 0;
118.
119.     //INICIO DE ACUMULACIÓN DE DATOS HISTÓRICOS
120.     //IMPORTANTE: MODIFICAR ESTOS VALORES SI SE RESETEA CÓDIGO
   Y NO SE QUIERE REINICIAR LA CUENTA
121.
122.       GENERACION = 0;
123.       CONSUMO = 0;
124.       AUTOCONSUMO = 0;
125.
126.       pinMode(rele, OUTPUT);
127.     }
128.
129.
130.
131.     void loop() {
132.
133.     //DECLARACIÓN DE LOS CONTADORES
134.      tiempototal = tiempo;
135.      tiempo = millis();
136.
137.
138.     //////////MEDIDAS AC: SALIDA DEL INVERSOR Y DE LA
   ACOMETIDA//////////
139.     //VOLTAJE AC
140.
141.       float Vinv, Vred =get_voltage(); //Voltage eficaz (V-RMS)
142.
143.       Serial.print("Vinv: ");
144.       Serial.print(Vinv,3);
145.       Serial.println(" VAC");
146.
147.       Serial.print("Vred: ");
148.       Serial.print(Vred,3);
149.       Serial.println(" VAC");
150.
151.
152.     //CORRIENTE AC
```

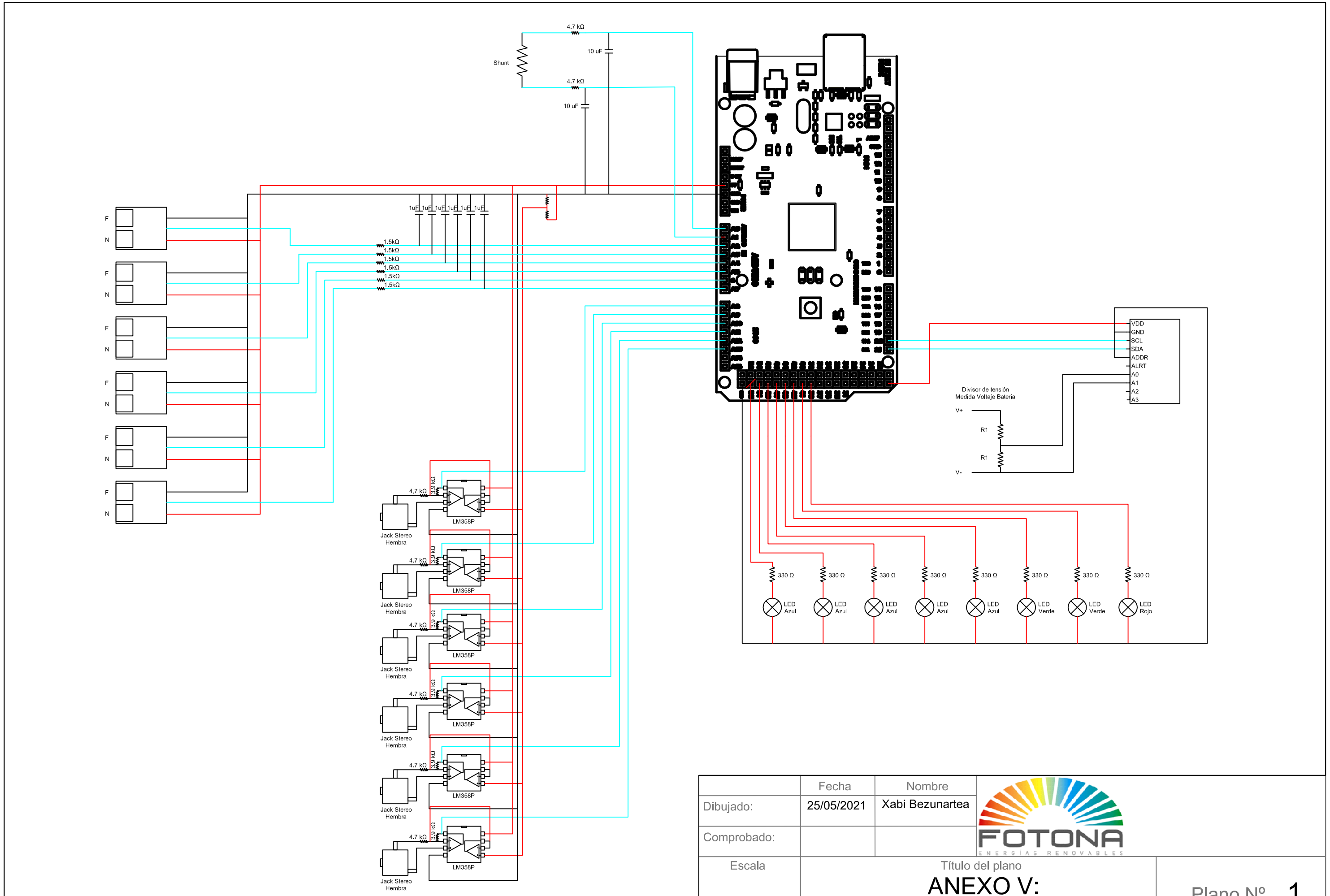
```
153.        double Iinv= emon1.calcIrms(1480);        // Cálculo de
    corriente rms I inversor
154.        double Ired = emon2.calcIrms(1480);        // Cálculo de
    corriente rms I red
155.
156.        Serial.print("Iinv= ");
157.        Serial.print(Iinv);
158.        Serial.println(" A");
159.        Serial.print("Ired= ");
160.        Serial.print(Ired);
161.        Serial.println(" A");
162.
163.      ////CALCULOS
164.
165.        tiempoproceso= tiempo-tiempototal;
166.        float T= tiempoproceso/1000;             //Tiempo del loop
    en segundos
167.
168.        Serial.print("Tiempo proceso: ");
169.        Serial.print(tiempoproceso);
170.        Serial.println(" ms");
171.
172.
173.      //Potencia de salida del inversor
174.      float Pinv, Einv;
175.
176.        Pinv = Vinv*Iinv;                       //Se supone FP=1
177.
178.      //Potencia intercambiada con la red
179.      float Pred, Ered;
180.
181.        Pred = Vred*Ired;                       //Se supone FP=1
182.
183.      //GENERACIÓN
184.
185.       Pgen= Pinv;
186.
187.       Egen = Egen + Pgen*T/3600;
188.
189.      //IMPORTACIÓN, EXCEDENTES y CONSUMO
190.      //Si la potencia de la red es positiva, se está exportando
    energía (por criterio de signos)
191.      //Si l apotencia de la red es negativa, se está importando
    energía
192.        if (Pred >= 0){
193.          Pex = Pred;
194.          Pimp = 0;
195.        }
196.        else {
197.          Pimp = -Pred;      //Potencia importada en valor
    absoluto
198.          Pex = 0;
199.        }
200.
201.        Eex = Eex + Pex*T/3600;
202.        Eimp = Eimp + Pimp*T/3600;
203.
204.      //CONSUMO DE ENERGÍA
205.
206.        Pcons = Pgen - Pex + Pimp;
207.        Econs = Econs + Pcons*T/3600;
```

```
208.
209.      //AUTOCONSUMO
210.      //Cuando hay excedentes, la energía autoconsumida es la
   resta entre la generación y los excedentes
211.      //Cuando se importa energía la energía autoconsumida es
   toda la generada
212.
213.        if (Pex >0){
214.          Pautoc = Pgen - Pex;
215.        }
216.        else {
217.          Pautoc = Pgen;
218.        }
219.        Eautoc = Eautoc + Pautoc*T/3600;
220.
221.      //PORCENTAJE DE AUTOCONSUMO ACUMULADO
222.      //Generación total y consumo total acumulados
223.        GENERACION= GENERACION + Egen;
224.        CONSUMO = CONSUMO + Econs;
225.        AUTOCONSUMO = AUTOCONSUMO + Eautoc;
226.
227.        porcentaje_autoconsumo = (AUTOCONSUMO/GENERACION)*100;
228.        porcentaje_excedente = 100 - porcentaje_autoconsumo;
229.
230.        cobertura_demanda = (AUTOCONSUMO/CONSUMO)*100;
231.        porcentaje_importado = 100 - cobertura_demanda;
232.
233.
234.      //GESTIÓN DE EXCEDENTES
235.        if (Pex >0){
236.          digitalWrite(rele, HIGH); //Cuando hay excedentes se
   activa la salida del relé
237.        }
238.        else {
239.          digitalWrite(rele, LOW);
240.        }
241.      Serial.print("Estado del relé: ");
242.      Serial.println(Rele);
243.
244.       if (tiempototal > FrecuenciaEnvio){
245.        //ENVÍO DE LOS DATOS ALMACENADOS EN LOS ÚLTIMOS 5 MIN
246.        //Se envía Egen, Econs, Eimp, Eex, Eautoc,
   porcentaje_autoconsumo, porcentaje_excedentes, cobertura_demanda
247.
248.          sprintf(pageAdd,"/databaseconnect.php?lectura=%s",total
   Count);
249.
250.        //SI FALLA LA CONEXIÓN AVISA, SI NO SE LLAMA getPage
251.          if(!getPage(serverName,serverPort,pageAdd)) Serial.prin
   t(F("Fail "));
252.          else Serial.print(F("Pass "));
253.          totalCount++;
254.          Serial.println(totalCount,DEC);
255.
256.
257.        //RESETEO DE LOS VALORES DE ENERGÍA ACUMULADOS
258.        //Valores de energía calculados en el último periodo de
   toma de datos
259.        //Los valores totales no se resetean porque sirven para
   acumular datos históricos
260.        Egen =0;
```

```
261.        Eimp =0;
262.        Eex =0;
263.        Econs =0;
264.        Eautoc =0;
265.
266.
267.        //RESETEO DEL CONTADOR
268.        setMillis(nuevo_valor);
269.        tiempo = millis();
270.        tiempototal = tiempo;
271.       }
272.
273.     }
274.
275.
276.
277.
278.
279.     //CÁLCULO DEL VOLTAJE AC RMS
280.     //IMPORTANTE: NECESITA 5V COMO REFERENCIA ANALÓGICA
281.     //DEVUELVE EL VOLTAJE DE SALIDA DEL INVERSOR Y EL VOLTAJE
   DE SALIDA DE LA ACOMETIDA
282.     float get_voltage(void)
283.     {
284.       //analogReference(DEFAULT); //POR SI SE HA CAMBIADO EN
   OTRO MOMENTO
285.       float adc_inv, adc_red;
286.       float volt_inst=0;
287.       float volt_inst_red=0;
288.       float Sumatoria=0;
289.       float Sum=0;
290.       float V_inv, V_red;
291.       long tiempo_init=millis();
292.       int N=0;
293.
294.       while( (millis() - tiempo_init) < 600)//Duración 0.6
   segundos(Aprox. 30 ciclos de 50Hz)
295.       {
296.
297.         adc_inv = analogRead(A3);    //Entrada sensor de
   voltaje del inversor
298.         adc_red = analogRead(A4);    //Entrada sensor de
   voltaje de la red
299.
300.         volt_inst = map(adc_inv,adc_min,adc_max,volt_multi_n,vo
   lt_multi_p);
301.         volt_inst_red = map(adc_red,adc_min_red,adc_max_red,vol
   t_multi_n_red,volt_multi_p_red);
302.
303.         Sumatoria = Sumatoria+sq(volt_inst);    //Sumatoria de
   Cuadrados
304.         Sum = Sum+sq(volt_inst_red);            //Sumatoria de
   Cuadrados
305.
306.         N = N+1;
307.         delay(1);
308.       }
309.
310.       V_inv=sqrt((Sumatoria)/N);  //ecuación del RMS
311.       V_red=sqrt((Sum)/N);        //ecuación del RMS
312.
```

```
313.       return V_inv, V_red;
314.     }
315.
316.     //RESETEO DEL CONTADOR para que no se desborde
317.     void setMillis(unsigned long nuevo_valor){
318.       uint8_t oldSREG = SREG;
319.       cli();
320.       timer0_millis = nuevo_valor;
321.       SREG = oldSREG;
322.     }
323.
324.
325.     //ENVÍO DE LOS DATOS
326.     byte getPage (char* domainBuffer,int thisPort,char* page)
327.     {
328.       int inChar;
329.       char outBuf[64];
330.
331.       Serial.print(F("connecting..."));
332.
333.
334.       if(client.connect(domainBuffer,thisPort) == 1)
335.       {
336.         Serial.print(F("connected to "));
337.         Serial.println(domainBuffer);
338.         Serial.println();
339.
340.         Serial.println(Egen);
341.         Serial.println(Econs);
342.         Serial.println(Eautoc);
343.         Serial.println(Eex);
344.         Serial.println(Eimp);
345.
346.         client.print("GET /databaseconnect.php?generacion=");
347.         client.print(Egen);
348.         client.print("&consumo=");
349.         client.print(Econs);
350.         client.print("&autoconsumo=");
351.         client.print(Eautoc);
352.         client.print("&exportado=");
353.         client.print(Eex);
354.         client.print("&importado=");
355.         client.print(Eimp);
356.         client.print("&porcentaje_autoconsumo=");
357.         client.print(porcentaje_autoconsumo);
358.         client.print("&porcentaje_excedente=");
359.         client.print(porcentaje_excedente);
360.         client.print("&cobertura_demanda=");
361.         client.print(cobertura_demanda);
362.         client.print("&porcentaje_importado=");
363.         client.print(porcentaje_importado);
364.         client.println(" HTTP/1.1");
365.
366.         sprintf(outBuf,"Host: %s",serverName);
367.         client.println(outBuf);
368.         client.println(F("Connection: close\r\n"));
369.
370.       }
371.       else
372.       {
373.         Serial.println(F("failed"));
```

```
374.         return 0;
375.       }
376.
377.       int connectLoop = 0;
378.
379.       while(client.connected())
380.       {
381.         while(client.available())
382.         {
383.           inChar = client.read();
384.           Serial.write(inChar);
385.           connectLoop = 0;
386.         }
387.
388.         delay(1);
389.         connectLoop++;
390.         if(connectLoop > 10000)
391.         {
392.           Serial.println();
393.           Serial.println(F("Timeout"));
394.           client.stop();
395.         }
396.       }
397.
398.       Serial.println();
399.       Serial.println(F("disconnecting."));
400.       client.stop();
401.       return 1;
402.     }
```



	Fecha	Nombre
Dibujado:	25/05/2021	Xabi Bezunartea
Comprobado:		

FOTONA ENERGÍAS RENOVABLES

Escala: S/E

Título del plano: ANEXO V: Montaje del dispositivo

Plano Nº 1